Introduction to Solend

The autonomous interest rate machine for Solana

Solend is an algorithmic, decentralized protocol for lending and borrowing on Solana.

Lending and borrowing has proven itself as key in a DeFi ecosystem. However, current products are slow and expensive. On Solana, Solend can scale to being 100x faster and 100x cheaper. Solend aims to be the easiest to use and most secure solution on Solana.

With Solend, you can do the following:

- Earn interest
- Borrow
- Leverage long
- Short

Start Here (Desktop)

Install a Solana Wallet

In order to use Solend, you will need a Solana wallet. We recommend Phantom.

Get SOL for Gas

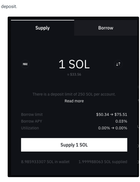
Actions on Solend require SOL as gas. You can buy SOL and transfer it to your account from exchanges such as FTX. After that, you're all set to make your first deposit!

Choose an asset to deposit. Major ones like USDC, ETH, BTC or SOL are a good start.

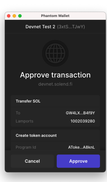
Deposit

Select an asset to deposit. You will be receiving a yield (Supply APY) on the asset you deposited.

Enter amount to deposit.



Approve the transaction in your wallet.



Borrow

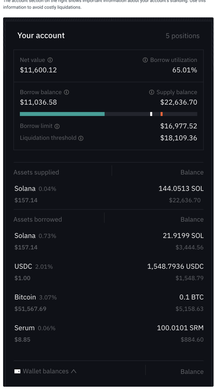
Once you've deposited funds, you can use them as collateral for a loan.

Specify how much you want to borrow.

Approve the transaction in your wallet again.

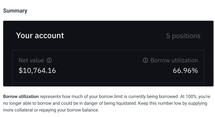
Account Panel

The account section on the right shows important information about your account's standing. Use this information to avoid costly liquidations.



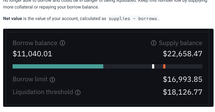
Let's go through each section from top to bottom.

Summary



Borrow utilization represents how much of your borrow limit is currently being borrowed. At 100%, you're no longer able to borrow and could be in danger of being liquidated. Keep this number low by supplying more collateral or repaying your borrow balance.

Net value is the value of your account, calculated as `supplies - borrows`.



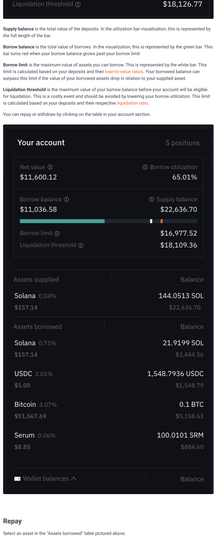
Supply balance is the total value of the deposits. In the utilization bar visualization, this is represented by the full length of the bar.

Borrow balance is the total value of borrows. In the visualization, this is represented by the green bar. This bar turns red when your borrow balance grows past your borrow limit.

Borrow limit is the maximum value of assets you can borrow. This is represented by the white bar. This limit is calculated based on your deposits and their loan-to-value ratios. Your borrowed balance can surpass this limit if the value of your borrowed assets drop in relation to your supplied asset.

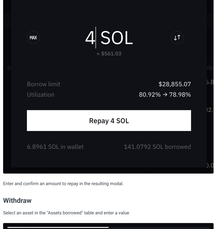
Liquidation threshold is the maximum value of your borrow balance before your account will be eligible for liquidation. This is a costly event and should be avoided by lowering your borrow utilization. This limit is calculated based on your deposits and their respective liquidation ratio.

You can repay or withdraw by clicking on the table in your account section.



Repay

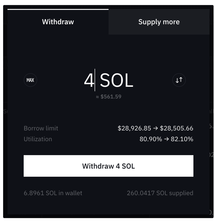
Select an asset in the 'Assets borrowed' table pictured above.



Enter and confirm an amount to repay in the resulting modal.

Withdraw

Select an asset in the 'Assets borrowed' table and enter a value



Enter and confirm an amount to withdraw in the resulting modal.

Ledger

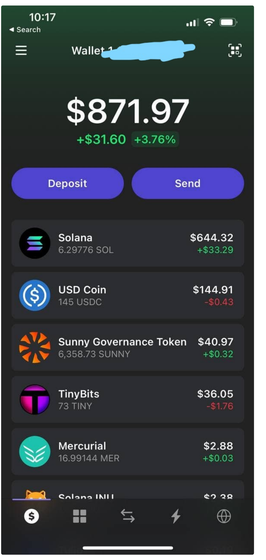
In order to use your Ledger device with Solend (and other Solana dapps), you must enable blind signing.

1. Connect and unlock your Ledger device.
2. Open the **Solana** app.
3. Press the right button to navigate to **Settings**. Then press both buttons to validate.
4. In the **Blind signing** settings, select **Yes** by pressing both buttons.

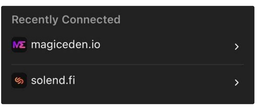
Start Here (Mobile)

Our site is very well mobile-optimised! Check out the future of finance, anywhere and anytime. Simply follow these instructions to get started.

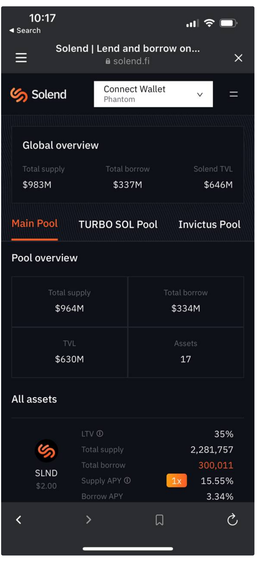
Install the latest Phantom wallet (currently only on iOS here). Set it up as normal, or log in to your existing Phantom wallet account. Sweet Phantom UI looks something like this.



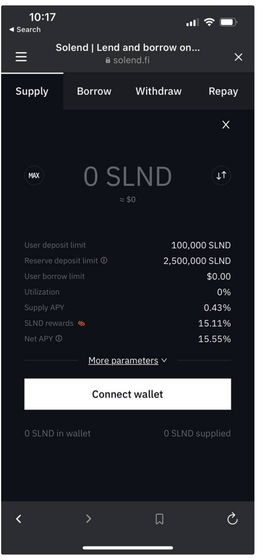
In the bottom right corner, click the Global web icon. This lets you browse websites to connect your wallet to. Go to https://solend.fi, our site.



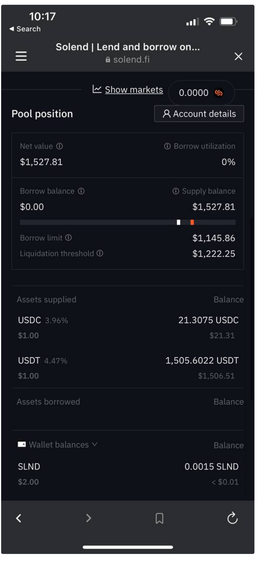
Solend will show up on Recently Connected in the future. Connect your Phantom to our UI in the top right hand corner.



You can now view the various assets, and supply/borrow against them by selecting them on this screen.



Check out your account details by clicking on "Show Account" at the top to see this screen, which details your current supply/borrows, and etc.



Supply & Borrow APY

Supply APY

Users who supply collect an APY from the users who borrow. The borrow APY is split across the entire pool, so Supply APY = Borrow APY * Utilization. For example, if there is 2 BTC in a pool, and 1 BTC is lent out at 20% APY, the suppliers of the 2 BTC will receive 10% APY each.

This APY is given in the **same token as the supply**. Deposit **BTC** for **BTC** yields, or **SOL** for **SOL** yields. This is because the APY on borrows is charged on the same yield as well. The displayed APY is based on compounding over a year.

Each Token has a different set of Parameters that determines their actual yields at different points, and can be seen here.

Additionally, you get Liquidity Mining (LM) rewards on SLND, UST, stSOL and mSOL. These LM rewards are collectible on the 3rd of every month, at UTC Midnight.

Borrow APY

When you are borrowing, you are **paying** a Borrow APY to the pool (for the users who supplied). This borrow APY is calculated based on the Parameters & The Math, and are displayed on our UI when you borrow. Borrow APY is added to your loan on a per-slot basis, so the amount of money you have to repay goes up over time.

Get paid to borrow!

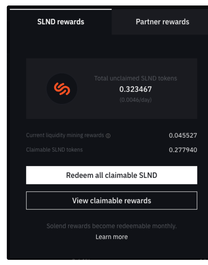
On most of the tokens, such as USDC and SOL, borrowers also receive SLND rewards for borrowing. These are claimable on the 3rd of every month at UTC Midnight, same as Liquidity Mining. For these, you can even get paid to borrow, as SLND Rewards > Borrow APY. These are displayed on our UI as a negative APY.

There is no "Deadline" for you to repay your loans. They will just tick up over time, but you can repay them any time!

Liquidity Mining Rewards

Viewing Rewards

Rewards can be viewed by clicking on the Pending Rewards button on the top right, near the Connect Wallet button.



Claiming Rewards

Rewards are claimable on the 3rd of each month, at UTC Midnight.

SOCN rewards for the month will be claimable once the token launches.

LM 2.0

In addition to the standard LM program where users receive SLND tokens, we are currently running a trial on a LM 2.0 program that involves call options.

An additional 10% of SLND tokens have been earmarked for LM rewards with the following parameters:

Options are American Strike

Price: 50% of the SLND price at the end of March

Expiry: One month

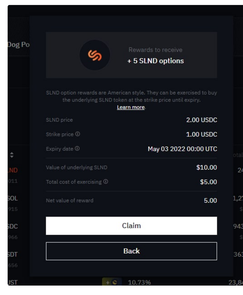
These details are subject to change, but the terms are as follows:

SLND price is at $2.00 on March 31st, you get to claim call options with a strike at $1.50 on April 3rd. You can then exercise them for the token, paying the strike to receive the full token. Exercising the option will involve topping up $1.00 per SLND token ($1.00 given to you + $1.00 your entry price). You can choose to hold or sell the SLND token after that.

Again, details are subject to change, and will eventually replace our regular LM program.

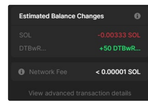
Claiming & Exercising LM Options

On the 3rd of every month at UTC 00:00, your options become claimable alongside your regular LM rewards.



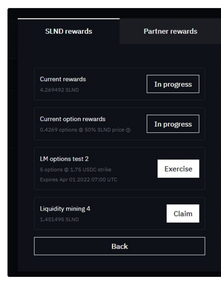
Currently, the parameters are set at 50% ITM. Thus, assuming a $2 SLND price at the end of March, the strike price of the option will be $1 USDC. This means that you can pay 1 USDC to receive 1 SLND token worth $2. The image above shows a SLND LM reward of 5 SLND options, worth $5.

The expiry date shown is in real time, so it will expire exactly at the stroke of midnight according to the expiry date displayed (not slot times).

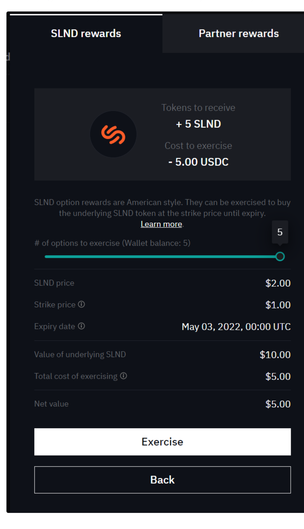


1 Option Token = 0.1 SLND token, so 5 SLND will be 50 Options as displayed by Phantom's UI. Thus, you can claim your LM rewards in denominations of 0.1. If you have <0.1 SLND, they will be rolled forward to the next month. Our UI will handle this, so no action is needed from you.

When you're ready, click on the Exercise button shown below to open the exercise UI.



On our exercise UI, you can swap your USDC for SLND immediately without leaving our site.



Liquidations

Your current "Borrow Balance" and "Supply Balance" are calculated using our 2 oracles, Pyth (main) and Switchboard (backup). Using these oracles, we can calculate your current health factor.

When your Health Bar reaches the orange line, your account is open to being liquidated. This orange line, the Liquidation Threshold, is based on the parameters set on different tokens. We calculate this liquidation threshold based on the weighted average of all the assets you deposited.

This can be found in your Account Details screen

When your account is open to being liquidated, our Third Party Liquidators will repay **20%** of your loans by selling the equivalent amount in Collateral. The liquidators will also collect an additional **5%** on the amount (20%) they liquidated, as a bounty to secure our protocol. You can learn more about becoming a liquidator here.

Liquidation Example

Let's say Bob supplied $10,000 USDC and borrowed $7,500 of BTC. BTC then goes up by 6.69% putting the BTC value at ~$8,000, making Bobs account eligible for liquidation.

The liquidator repays 20% of the BTC loan, $1,600, collects $1,600 from the collateral USDC supply to cover the BTC, then collects an additional 5% ($80) as a bonus for completing the liquidation successfully.

Now Bob has $10,000 - $1,600 - $80 = $8,320 in USDC, and $8,000 - $1,600 = $6,400 in BTC borrows while the liquidator paid $1600 in BTC and received 1,680 in USDC, for a profit of $80.

Bob's Liquidation Threshold drops from 80% -> 77%.

Risks

All DeFi protocols, including Solend, come with risks, which are important to understand before depositing significant amounts of crypto. The main risks involved in using Solend are outlined here.

Smart Contract Risk

This is a risk that the Solend smart contracts get exploited to steal or permanently freeze funds. This risk is inherent to all smart contracts and can never be fully eliminated, but can be mitigated in various ways.

- Solend has undergone audits by Kudelski, Neodyme, and OSEC.
- Solend has a bug bounty program which will pay up to $1MM if a critical vulnerability is found, in order to encourage responsible disclosure rather than hacking.

The Solend smart contracts have been live on mainnet with hundreds of millions in total value locked (TVL) since 2021.

100% Utilization Risk

When an asset is fully utilized (100% of supply is lent out), there are no tokens left in the pool, which means that withdrawals and borrows will fail. Users have to wait until the utilization rate goes down, either through some users repaying their loans or depositing new funds.

A user is more likely to be affected by this if their deposit represents a large share of the pool, or if the asset has extremely high borrow demand.

Oracle Risk

Solend relies on Pyth and Switchboard for their price feeds to power liquidations. There is a risk that these oracles report incorrect prices, causing wrongful liquidations.

Liquidation Risk

Solend offers over-collateralized loans, which means loans must be backed by collateral of greater value than the loan. If the value of the collateral dips below a threshold (determined by asset LTVs), a user's position will be liquidated with a liquidation penalty.

Untimely Liquidation Risk

In the event of large-scale liquidations or market turmoil, there is a possibility that assets liquidated are unable to cover the loans taken out by the liquidated user. The shortfall "or negative balance" is treated as "bad debts". Solend manages this proactively by isolating newer or riskier tokens in Isolated Pools and managing deposit limits or collateralization ratios to keep the protocol safe.

In the past, Bad debt in main pool has been filled via top-ups from the Insurance fund, which can be found at our DAO addresses.

FAQ

FAQ

Does Solend have a token?

Yes. SLND is a spl-token on here.

What was the IDO price?

$6.57

What oracles does Solend use?

Solend uses Pyth, and Switchboard as a backup.

When will Solend list X token?

Since Solend uses Pyth and Switchboard, it needs a feed from one in order to list an asset.

You can suggest tokens for isolated pools in Solend Discord.

Why are fees a couple dollars?

TL;DR it's only expensive the first time you use Solend.

Solana state accounts need to be created to store state about your positions on Solend. Some SOL is required to make these accounts rent-exempt. These fees are only paid once, the first time you interact with Solend. Read more.

What is Nope?

Nope Finance is a project that Solend acquired in June. Nope Finance already had the NOPE token, so Solend offered a conversion to bring the Nope community into Solend's.

Conversion can be done by following this guide.

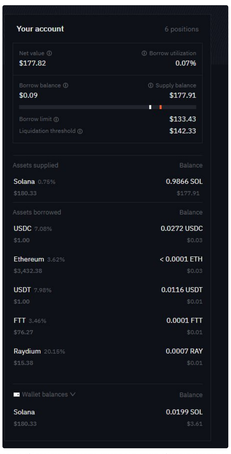
To summarize, visit a Serum DEX client (e.g. Bonfida) and manually add the SLND/NOPE market. The market ID is `L48mQEKqj4f9BAcokgbB2vZ29w8X8HaYS3njco5JfRH`. The conversion rate is 346.15038 NOPE to 1 SLND. Details of this announcement are here.

Debugging FAQ

Debugging FAQ

Transaction too large: 1317 > 1232

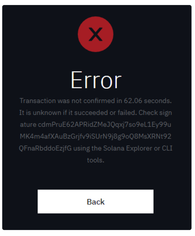
Usually caused by having too many positions and trying to operate on SOL which uses WSOL under the hood. The extra work that needs to be done causes the transaction to be too large.



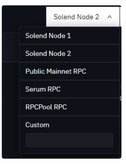
User has a total of 6 positions. Even though they have a tiny amount of borrows, each position counts.

Fix: Close one of your positions. This can be done by *fully* repaying one of your borrows, or fully withdrawing one of your deposits (other than SOL). You may have to buy some tokens to repay the tiny amount of debt for a borrow. Then try withdrawing SOL again.

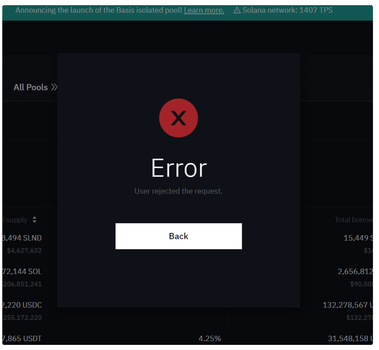
Error: Transaction was not confirmed in >60 seconds. It is unknown if it succeeded or failed.



Confirm whether the transaction went through by using a Solana Explorer, either SolanaFM or Solscan to check whether the transaction went through. Alternatively, you can wait for a few minutes to see if your transaction is reflected on the platform. If not, try switching RPC Nodes.

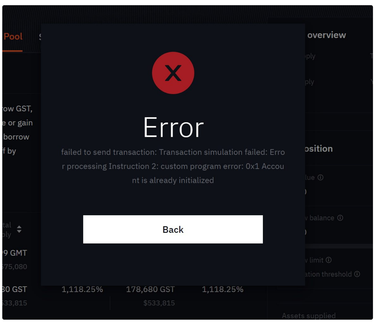


Error: User rejected the request



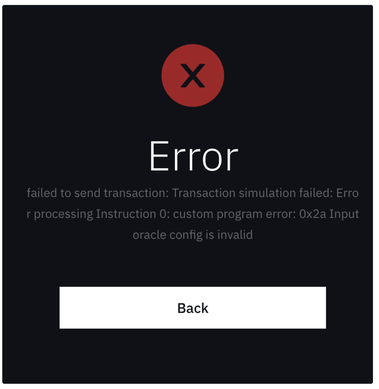
Usually a ledger issue or RPC issues, refresh and try again.

Error: Account is already initialized



Caused by a lack of SOL / unable to pay rent for account creation. Ensure that the wallet has 0.1 SOL.

Error: Input Oracle Config

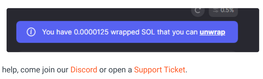


This happens when Switchboard/Pyth is not pushing price updates, usually due to network congestion. Try again later when TPS is higher, or flag the situation in Discord.

Error: SOL disappeared

If a transaction fails, you might get wrapped SOL instead of SOL. This may look like your SOL disappeared, simply go to Jupiter to swap wSOL back into SOL.

It will look like this:



If you still need help, come join our Discord or open a Support Ticket.

DAO

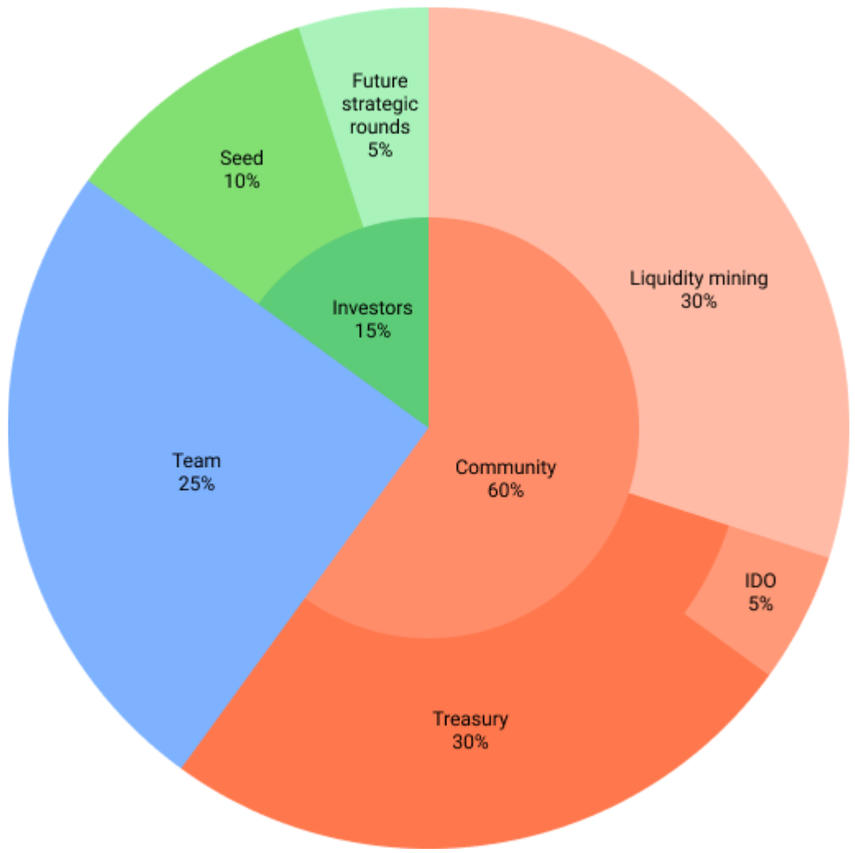
DAO Wallets

Description	Address
fee receiver	`9RuqAN42PTUi9ya59k9suGATrkqzvb9gk2QABJtQzGP5`
orca LPs	`AHcFLPeD964ucZ4F4b3ikPNHDPp8kz1ZmW7nid7r1hBQ`
other treasury operations	`GDmSxpPzLkfxxr6dHLNRnCoYVGzvgc41tozkrr4pHTjB`
treasury msig	`EaFPY9LTQeFR7SEyfbKKFuVMtYvUBbYiiK7WvBJJ7iBU`
LM msig	`5QbRL9MU5QakL5Fx2He9YaiUzB3TQpVAUBR2ARKN1NrM`
gnosis safe (eth)	`0xA21a60651183c11eCa1E6aA03B2cEBbe49Ca1a4E`
treasury operations (eth)	`0xF41f7FebD6574FE0B08B999598df44f6B3E95006`
treasury msig (old)	`BX1VZoS4sDyBeXWuLzfbngYtdJGEEBK39PzUGfWYpFNu`

These wallets belong to the DAO and will be governed by SLND holders.

Token

There are 100M SLND tokens. SLND's distribution is as follows:



SLND token distribution. Note that IDO allocation is part of the treasury.

60% of SLND is allocated to the community. Half of that is allocated to the liquidity mining program and the other half is allocated to the Solend Treasury, owned and governed by the Solend DAO. 5% of SLND is allocated to the IDO, coming from the treasury. The Solend Treasury will own the IDO funds and LP position (details in IDO section.)

25% of SLND is allocated to the core team.

15% of SLND is allocated to VCs and Angels. Only 10% was distributed in the seed round, but an additional 5% is set aside for a potential future raise in case it's needed.

Lockups

Seed round participants have a 3-year unlock schedule with the first third unlocking October 1, 2022, and the rest unlock 1/36th monthly after. The team also has a 3-year unlocking schedule with the first third unlocking June 1, 2022 or later (based on join date). There is no lockup for IDO participants.

Circulating supply

Solend's liquidity mining program is the main driver of circulating supply, especially early on. See this spreadsheet for a projection of the SLND circulating supply.

IDO

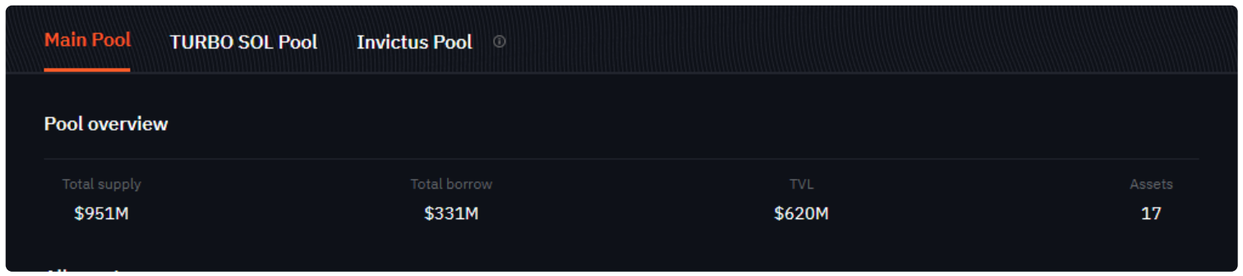
Read the IDO announcement here:

Fundraise, tokenomics, and IDO
Medium

IDO is available at ido.solend.fi and will be live on Nov 1, 2021 at 12:00pm UTC. There's also a backup at ido-ui.pages.dev.

The IDO UI is open source and available at https://github.com/solendprotocol/ido-ui

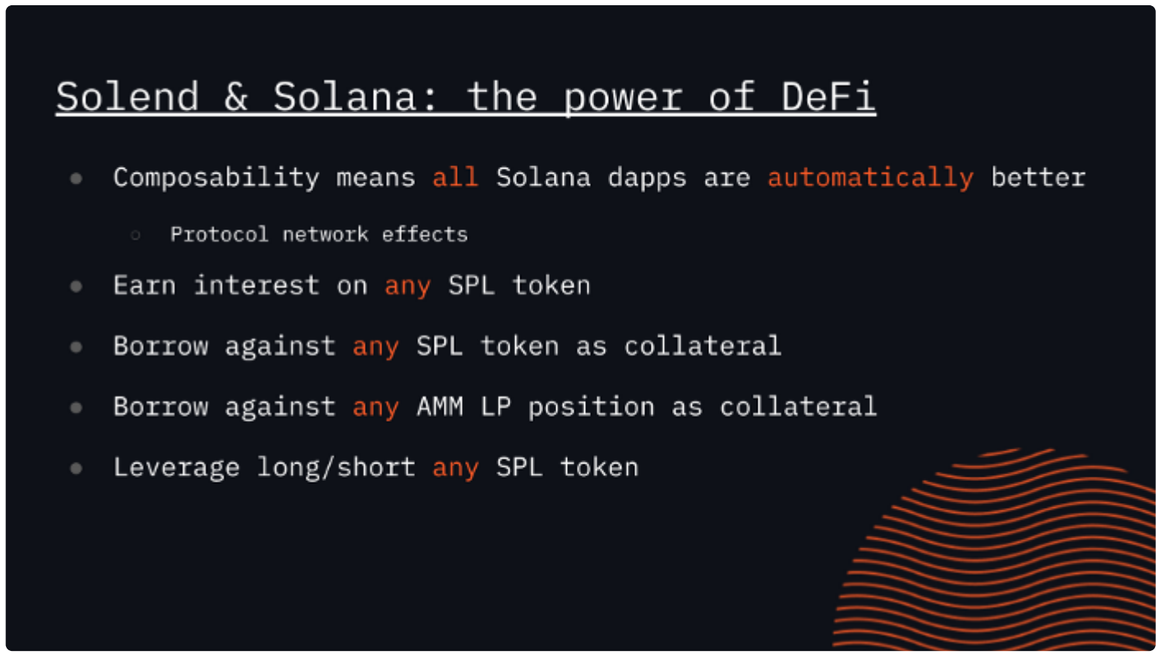
Solend Pools



The majority of Solend TVL sits in a global main pool and a series of smaller isolated pools and permissionless pools.

The global main pool can be accessed on the main page here, and follow the set parameters (variables set for each token).

Tokens can be listed in the main pool if they do not represent an increase in risk, mainly by having reliable oracles or thick liquidity to assist in liquidations. Most tokens will be listed on Isolated pools first before proceeding to Main pool after sufficient time.



Isolated pools are smaller pools that are used to list slightly riskier tokens such as Locked Staked IN (lsIN) or future tokens like xSTEP or SAMO. As these tokens have less liquidity and are more volatile, they might be an exploitable opportunity or a risk for the main pool, isolated pools are used to screen/quarantine them before adding them to the main pool.

This way, any exploit or risk will only affect the smaller TVLs present in the isolated pools instead of risking the bulk of TVL in the main pool.

Permissionless pools allow anyone to create an isolated pool on Solend. The creator decides and earns 20% of origination fees generated in that pool. Therefore, a successful Permissionless Pool means success for the pool creator.

There will be many more isolated pools to come. If you're interested in a particular one, bring it up on Discord, or come join our #permissionless-pools channel in Discord to talk about new pool ideas!

Parameters

APIs: When in doubt, follow the data here.

This config also includes oracles used, their addresses, refresh rates, and is better for integration.

Parameters:

Each reserve in Main or Isolated pool has these parameters that balance the supply & demand of the pool or governs the size of the loan you can take out (based on the quality and liquidity of the collateral asset).

The critical thing to remember is the borrow APR changes based on utilization. Utilization is Tokens Borrowed / Tokens Supplied. At 0% utilization, there is excess supply and no demand, while at 100% there is excess demand but no supply. Details are in Protocol Math & Fees.

Parameter	Value
Optimal utilization rate	Utilization where borrow APR starts to increase exponentially
Loan to value ratio	Borrow X% against your collateral
Liquidation penalty	% of your liquidated collateral to incentivize liquidators
Liquidation threshold	Account health that will trigger a liquidation
Min borrow APR	Borrow APR when Utilization = 0%
Optimal borrow APR	Borrow APR when at optimal utilization
Max borrow APR	Borrow APR when utilization = 100%
Borrow fee	Origination fee Solend keeps when you borrow
Flash loan fee	Origination fee Solend keeps when you borrow via flash loan
Host fee percentage	% of borrow fee kept by referral, UI or pool creator
Deposit limit	The cap of tokens that can be deposited into the pool
Borrow limit	The cap of tokens that can be borrowed into the pool

Recovery Mode:

Recovery Mode is triggered during times of heightened volatility or network congestion. During Recovery Mode, a council of contributors can make parameter changes with greater flexibility to minimize bad debt and damage to Solend and its users.

Note: These changes might lead to liquidations occurring, done so without liquidation penalty in market turmoil. These changes are made to reduce bad debt for Solend as a whole, and may lead to user positions being forced closed by liquidators without penalty.

When Recovery Mode triggers, announcements will be made on Twitter and in Solend's Discord. A Banner will also be added on the primary Solend UI at:https://solend.fi.

Main Pool

Solend's main TVL sits in our main pool. These parameters are set with protocol health in mind, and we avoid listing riskier assets here until they have strong liquidity or represent low security risk. This pool will contain assets we are comfortable with listing alongside our main TVL, and usually have more liquidity.

SOL

Parameter	Value
Optimal utilization rate (%)	80%
Loan to value ratio	75%
Liquidation penalty	5%
Liquidation threshold	85%
Min borrow APR	0%
Optimal borrow APR	8%
Max borrow APR	100%
Interest Rate Spread	10%
Borrow fee	0.1%
Flash loan fee	0.3%
Host fee percentage	20%
Deposit limit	5,000,000 SOL
Borrow limit	2,000,000 SOL

USDC

Parameter	Value
Optimal utilization rate	80%
Loan to value ratio	75%
Liquidation penalty	5%
Liquidation threshold	85%
Min borrow APR	0%
Optimal borrow APR	8%
Max borrow APR	200%
Interest Rate Spread	10%
Borrow fee	0.1%
Flash loan fee	0.3%
Host fee percentage	20%
Deposit limit	150,000,000 USDC
Borrow limit	80,000,000 USDC

soETH (Sollet) (Deprecating)

Migrate here with this guide.

Parameter	Value
Optimal utilization rate	80%
Loan to value ratio	75%
Liquidation penalty	5%
Liquidation threshold	85%
Min borrow APR	0%
Optimal borrow APR	8%
Max borrow APR	100%
Borrow fee	0.1%
Flash loan fee	0.3%
Host fee percentage	20%
Deposit limit	0 soETH
Borrow limit	0 soETH

ON THIS PAGE

SOL
USDC
soETH (Sollet) (Deprecating)
BTC (Sollet)
SRM
USDT
FTT (Wormhole)
soFTT (Sollet) [deprecating]
RAY
SBR (Deprecating)
MER (Deprecating)
mSOL
ETH (Wormhole)
SLND
scnSOL
stSOL
UST (Wormhole) (Deprecating)
ORCA
Saber USDT-USDC LP
Saber mSOL-SOL LP

BTC (Sollet)

Parameter	Value
Optimal utilization rate	80%
Loan to value ratio	75%
Liquidation penalty	5%
Liquidation threshold	85%
Min borrow APR	0%
Optimal borrow APR	8%
Max borrow APR	100%
Interest Rate Spread	10%
Borrow fee	0.1%
Flash loan fee	0.3%
Host fee percentage	20%
Deposit limit	2,500 BTC
Borrow limit	None

SRM

Parameter	Value
Optimal utilization rate	80%
Loan to value ratio	65%
Liquidation penalty	5%
Liquidation threshold	75%
Min borrow APR	0%
Optimal borrow APR	12%
Max borrow APR	150%
Interest Rate Spread	15%
Borrow fee	0.1%
Flash loan fee	0.3%
Host fee percentage	20%
Deposit limit	750,000 SRM
Borrow limit	None

USDT

Parameter	Value
Optimal utilization rate	80%
Loan to value ratio	75%
Liquidation penalty	5%
Liquidation threshold	85%
Min borrow APR	0%
Optimal borrow APR	8%
Max borrow APR	50%
Interest Rate Spread	10%
Borrow fee	0.1%
Flash loan fee	0.3%
Host fee percentage	20%
Deposit limit	40,000,000 USDT
Borrow limit	36,000,000 USDT

FTT (Wormhole)

Parameter	Value
Optimal utilization rate	80%
Loan to value ratio	65%

Liquidation penalty	5%
Liquidation threshold	75%
Min borrow APR	0%
Optimal borrow APR	12%
Max borrow APR	150%
Interest Rate Spread	15%
Borrow fee	0.1%
Flash loan fee	0.3%
Host fee percentage	20%
Deposit limit	300,000 FTT
Borrow limit	None

soFTT (Sollet) [deprecating]

Migrate here with this guide.

Parameter	Value
Optimal utilization rate	80%
Loan to value ratio	75%
Liquidation penalty	5%
Liquidation threshold	80%
Min borrow APR	0%
Optimal borrow APR	12%
Max borrow APR	150%
Interest Rate Spread	15%
Borrow fee	0.1%
Flash loan fee	0.3%
Host fee percentage	80%
Deposit limit	0 FTT
Borrow limit	0 FTT

RAY

Parameter	Value
Optimal utilization rate	80%
Loan to value ratio	65%
Liquidation penalty	5%
Liquidation threshold	85%
Min borrow APR	0%
Optimal borrow APR	30%
Max borrow APR	200%
Interest Rate Spread	15%
Borrow fee	0.1%
Flash loan fee	0.3%
Host fee percentage	20%
Deposit limit	1,250,000 RAY
Borrow limit	None

SBR (Deprecating)

Migrating to an Isolated Pool to help Main pool become safer, allowing Main parameters to be increased.

Parameter	Value
Optimal utilization rate	80%
Loan to value ratio	0%
Liquidation penalty	20%
Liquidation threshold	50%

Min borrow APR	0%
Optimal borrow APR	12%
Max borrow APR	200%
Borrow fee	0.1%
Flash loan fee	0.3%
Host fee percentage	20%
Deposit limit	0 SBR
Borrow limit	0 SBR

MER (Deprecating)

Migrating to an Isolated Pool to help Main pool become safer, allowing Main parameters to be increased.

Parameter	Value
Optimal utilization rate	80%
Loan to value ratio	0%
Liquidation penalty	20%
Liquidation threshold	50%
Min borrow APR	0%
Optimal borrow APR	30%
Max borrow APR	200%
Borrow fee	0.1%
Flash loan fee	0.3%
Host fee percentage	20%
Deposit limit	0 MER
Borrow limit	0 MER

mSOL

Parameter	Value
Optimal utilization rate	80%
Loan to value ratio	75%
Liquidation penalty	5%
Liquidation threshold	85%
Min borrow APR	0%
Optimal borrow APR	30%
Max borrow APR	200%
Interest Rate Spread	10%
Borrow fee	0.1%
Flash loan fee	0.3%
Host fee percentage	20%
Deposit limit	2,500,000 mSOL
Borrow limit	None

ETH (Wormhole)

Parameter	Value
Optimal utilization rate	80%
Loan to value ratio	75%
Liquidation penalty	5%
Liquidation threshold	85%
Min borrow APR	0%
Optimal borrow APR	8%
Max borrow APR	100%
Interest Rate Spread	10%
Borrow fee	0.1%
Flash loan fee	0.3%

Host fee percentage	20%
Deposit limit	25,000 ETH
Borrow limit	None

SLND

Parameter	Value
Optimal utilization rate	80%
Loan to value ratio	35%
Liquidation penalty	20%
Liquidation threshold	50%
Min borrow APR	100%
Optimal borrow APR	250%
Max borrow APR	250%
Interest Rate Spread	20%
Borrow fee	5%
Flash loan fee	0.3%
Host fee percentage	20%
Deposit limit	5,000,000 SLND
User Deposit Limit	500,000 SLND
Borrow limit	500,000 SLND

scnSOL

Parameter	Value
Optimal utilization rate	80%
Loan to value ratio	75%
Liquidation penalty	5%
Liquidation threshold	85%
Min borrow APR	0%
Optimal borrow APR	12%
Max borrow APR	150%
Borrow fee	0.1%
Flash loan fee	0.3%
Host fee percentage	20%
Deposit limit	None
Borrow limit	None

stSOL

Parameter	Value
Optimal utilization rate	80%
Loan to value ratio	75%
Liquidation penalty	5%
Liquidation threshold	85%
Min borrow APR	0%
Optimal borrow APR	12%
Max borrow APR	150%
Interest Rate Spread	10%
Borrow fee	0.1%
Flash loan fee	0.3%
Host fee percentage	20%
Deposit limit	1,500,000 stSOL
Borrow limit	None

UST (Wormhole) (Deprecating)

Parameter	Value
Optimal utilization rate	80%
Loan to value ratio	0%
Liquidation penalty	0%
Liquidation threshold	50%
Min borrow APR	0%
Optimal borrow APR	12%
Max borrow APR	250%
Borrow fee	0.1%
Flash loan fee	0.3%
Host fee percentage	20%
Deposit limit	0 UST
User Deposit Limit	0 UST
Borrow limit	None

ORCA

Parameter	Value
Optimal utilization rate	80%
Loan to value ratio	35%
Liquidation penalty	20%
Liquidation threshold	50%
Min borrow APR	0%
Optimal borrow APR	30%
Max borrow APR	200%
Interest Rate Spread	20%
Borrow fee	0.1%
Flash loan fee	0.3%
Host fee percentage	20%
Deposit limit	750,000 ORCA
Borrow limit	None

Saber USDT-USDC LP

Parameter	Value
Optimal utilization rate	80%
Loan to value ratio	65%
Liquidation penalty	10%
Liquidation threshold	75%
Min borrow APR	0%
Optimal borrow APR	12%
Max borrow APR	100%
Borrow fee	0.1%
Flash loan fee	0.3%
Host fee percentage	20%
Deposit limit	500,000 USDT-USDC LP
Borrow limit	0 USDT-USDC LP

Saber mSOL-SOL LP

Parameter	Value
Optimal utilization rate	80%
Loan to value ratio	65%
Liquidation penalty	10%
Liquidation threshold	75%
Min borrow APR	0%
Optimal borrow APR	12%
Max borrow APR	100%
Borrow fee	0.1%
Flash loan fee	0.3%
Host fee percentage	20%
Deposit limit	25,000 mSOL-SOL LP
Borrow limit	0 mSOL-SOL LP

TURBO SOL Pool

What's TURBO SOL?

Out of all the assets in the main pool, USDC & SOL enjoy much better liquidity as they are used in a majority of liquidity pairs on Solana. Thus, we can push the LTVs and thresholds of SOL & USDC, allowing higher leverage strategies to be possible (20x leverage).

As a result, TURBO SOL is a pool with only these two highly liquid assets, with ultra degen settings. It is not possible to list other assets such as mSOL here, as they have more unreliable oracles or weaker liquidity.

What's the risk?

Due to the degen thresholds set, low price movements can easily liquidate users if they push the boundaries to max leverage.

Smart contract risk and other risks are minimal with this pool, as only the best quality assets (SOL/USDC) are used.

SOL

Parameter	Value
Optimal utilization rate (%)	90%
Loan to value ratio	90%
Liquidation penalty	3%
Liquidation threshold	95%
Min borrow APR	0%
Optimal borrow APR	10%
Max borrow APR	150%
Interest Rate Spread	10%
Borrow fee	0.1%
Flash loan fee	0.3%
Host fee percentage	20%
Deposit limit	100,000 SOL
Borrow limit	None

USDC

Parameter	Value
Optimal utilization rate (%)	90%
Loan to value ratio	90%
Liquidation penalty	3%
Liquidation threshold	95%
Min borrow APR	0%
Optimal borrow APR	15%
Max borrow APR	200%
Interest Rate Spread	10%
Borrow fee	0.2%
Flash loan fee	0.3%
Host fee percentage	20%
Deposit limit	20,000,000 USDC
Borrow limit	None

Step Pool

What's Step Pool?

Step pool offers xSTEP and STEP, allowing for multiple strategies such as borrowing STEP to deposit xSTEP, or depositing large amounts of STEP or xSTEP to unlock capital (SOL/USDC) against them.

What's the Risk?

There are some smart contract risks present due to the reliance on an external protocol. If Step is exploited and drained, STEP and xSTEP here would likely not get liquidated fast enough and cause socialized losses for lenders.

There is also the risk of oracle exploits, as xSTEP and STEP rely on Switchboard oracles to get up-to-date prices.

xSTEP

Parameter	Value
Optimal utilization rate (%)	80%
Loan to value ratio	50%
Liquidation penalty	10%
Liquidation threshold	65%
Min borrow APR	0%
Optimal borrow APR	30%
Max borrow APR	200%
Interest Rate Spread	20%
Borrow fee	1%
Flash loan fee	0.3%
Host fee percentage	20%
Deposit limit	25,000,000 xSTEP
User Deposit Limit	5,000,000 xSTEP
Borrow limit	None

STEP

Parameter	Value
Optimal utilization rate (%)	80%
Loan to value ratio	50%
Liquidation penalty	10%
Liquidation threshold	65%
Min borrow APR	0%
Optimal borrow APR	30%
Max borrow APR	200%
Interest Rate Spread	20%
Borrow fee	1%
Flash loan fee	0.3%
Host fee percentage	20%
Deposit limit	25,000,000 STEP
User Deposit Limit	5,000,000 STEP
Borrow limit	None

SOL

Parameter	Value
Optimal utilization rate (%)	80%
Loan to value ratio	75%
Liquidation penalty	5%
Liquidation threshold	80%
Min borrow APR	0%
Optimal borrow APR	8%
Max borrow APR	200%
Interest Rate Spread	20%
Borrow fee	0.5%
Flash loan fee	0.3%
Host fee percentage	20%
Deposit limit	50,000 SOL
User Deposit Limit	None
Borrow limit	None

USDC

Parameter	Value
Optimal utilization rate (%)	80%
Loan to value ratio	75%
Liquidation penalty	5%
Liquidation threshold	80%
Min borrow APR	0%
Optimal borrow APR	15%
Max borrow APR	200%
Interest Rate Spread	20%
Borrow fee	0.5%
Flash loan fee	0.3%
Host fee percentage	20%
Deposit limit	5,000,000 USDC
User Deposit Limit	None
Borrow limit	None

Bonfida Pool

Bonfida

Parameter	Value
Optimal utilization rate (%)	80%
Loan to value ratio	50%
Liquidation penalty	10%
Liquidation threshold	65%
Min borrow APR	0%
Optimal borrow APR	30%
Max borrow APR	200%
Interest Rate Spread	20%
Borrow fee	1%
Flash loan fee	0.3%
Host fee percentage	20%
Deposit limit	3,000,000 FIDA
User Deposit Limit	300,000 FIDA
Borrow limit	None

SOL

Parameter	Value
Optimal utilization rate (%)	80%
Loan to value ratio	75%
Liquidation penalty	5%
Liquidation threshold	80%
Min borrow APR	0%
Optimal borrow APR	8%
Max borrow APR	200%
Interest Rate Spread	20%
Borrow fee	0.5%
Flash loan fee	0.3%
Host fee percentage	20%
Deposit limit	50,000 SOL
User Deposit Limit	None
Borrow limit	None

USDC

Parameter	Value
Optimal utilization rate (%)	80%
Loan to value ratio	75%
Liquidation penalty	5%
Liquidation threshold	80%
Min borrow APR	0%
Optimal borrow APR	15%
Max borrow APR	200%
Interest Rate Spread	20%
Borrow fee	0.5%
Flash loan fee	0.3%
Host fee percentage	20%
Deposit limit	5,000,000 USDC
User Deposit Limit	None
Borrow limit	None

Star Atlas Pool

ATLAS

Parameter	Value
Optimal utilization rate (%)	80%
Loan to value ratio	50%
Liquidation penalty	10%
Liquidation threshold	65%
Min borrow APR	0%
Optimal borrow APR	30%
Max borrow APR	200%
Interest Rate Spread	20%
Borrow fee	1%
Flash loan fee	0.3%
Host fee percentage	20%
Deposit limit	200,000,000 ATLAS
User Deposit Limit	20,000,000 ATLAS
Borrow limit	None

POLIS

Parameter	Value
Optimal utilization rate (%)	80%
Loan to value ratio	50%
Liquidation penalty	10%
Liquidation threshold	65%
Min borrow APR	0%
Optimal borrow APR	30%
Max borrow APR	200%
Interest Rate Spread	20%
Borrow fee	1%
Flash loan fee	0.3%
Host fee percentage	20%
Deposit limit	3,000,000 POLIS
User Deposit Limit	300,000 POLIS
Borrow limit	None

USDC

Parameter	Value
Optimal utilization rate (%)	80%
Loan to value ratio	75%
Liquidation penalty	5%
Liquidation threshold	80%
Min borrow APR	0%
Optimal borrow APR	15%
Max borrow APR	200%
Interest Rate Spread	20%
Borrow fee	0.5%
Flash loan fee	0.3%
Host fee percentage	20%
Deposit limit	5,000,000 USDC
User Deposit Limit	None
Borrow limit	None

Dog Pool

SAMO

Parameter	Value
Optimal utilization rate (%)	80%
Loan to value ratio	50%
Liquidation penalty	10%
Liquidation threshold	65%
Min borrow APR	0%
Optimal borrow APR	30%
Max borrow APR	200%
Interest Rate Spread	20%
Borrow fee	1%
Flash loan fee	0.3%
Host fee percentage	20%
Deposit limit	250,000,000 SAMO
User Deposit Limit	25,000,000 SAMO
Borrow limit	None

USDC

Parameter	Value
Optimal utilization rate (%)	80%
Loan to value ratio	75%
Liquidation penalty	5%
Liquidation threshold	80%
Min borrow APR	0%
Optimal borrow APR	15%
Max borrow APR	200%
Interest Rate Spread	20%
Borrow fee	0.5%
Flash loan fee	0.3%
Host fee percentage	20%
Deposit limit	5,000,000 USDC
User Deposit Limit	None
Borrow limit	None

Stable Pool

PAI

Parameter	Value
Optimal utilization rate (%)	80%
Loan to value ratio	95%
Liquidation penalty	1%
Liquidation threshold	97%
Min borrow APR	0%
Optimal borrow APR	12%
Max borrow APR	200%
Interest Rate Spread	10%
Borrow fee	0.05%
Flash loan fee	0.3%
Host fee percentage	20%
Deposit limit	5,000,000 PAI
User Deposit Limit	5,000,000 PAI
Borrow limit	None

USDC

Parameter	Value
Optimal utilization rate (%)	80%
Loan to value ratio	95%
Liquidation penalty	1%
Liquidation threshold	97%
Min borrow APR	0%
Optimal borrow APR	12%
Max borrow APR	200%
Interest Rate Spread	10%
Borrow fee	0.05%
Flash loan fee	0.3%
Host fee percentage	20%
Deposit limit	25,000,000 USDC
User Deposit Limit	25,000,000 USDC
Borrow limit	None

USDT

Parameter	Value
Optimal utilization rate (%)	80%
Loan to value ratio	95%
Liquidation penalty	1%
Liquidation threshold	97%
Min borrow APR	0%
Optimal borrow APR	12%
Max borrow APR	200%
Interest Rate Spread	20%
Borrow fee	0.05%
Flash loan fee	0.3%
Host fee percentage	10%
Deposit limit	5,000,000 USDT
User Deposit Limit	5,000,000 USDT
Borrow limit	None

UST

Parameter	Value
Optimal utilization rate (%)	80%
Loan to value ratio	0%
Liquidation penalty	1%
Liquidation threshold	97%
Min borrow APR	0%
Optimal borrow APR	12%
Max borrow APR	200%
Interest Rate Spread	10%
Borrow fee	0.05%
Flash loan fee	0.3%
Host fee percentage	20%
Deposit limit	0 UST
User Deposit Limit	0 UST
Borrow limit	None

UXD

Parameter	Value
Optimal utilization rate (%)	80%
Loan to value ratio	95%
Liquidation penalty	1%
Liquidation threshold	97%
Min borrow APR	0%
Optimal borrow APR	12%
Max borrow APR	200%
Interest Rate Spread	10%
Borrow fee	0.05%
Flash loan fee	0.3%
Host fee percentage	20%
Deposit limit	25,000,000 UXD
User Deposit Limit	25,000,000 UXD
Borrow limit	None

USDH

Parameter	Value
Optimal utilization rate (%)	80%
Loan to value ratio	95%
Liquidation penalty	1%
Liquidation threshold	97%
Min borrow APR	0%
Optimal borrow APR	12%
Max borrow APR	200%
Interest Rate Spread	10%
Borrow fee	0.05%
Flash loan fee	0.3%
Host fee percentage	20%
Deposit limit	5,000,000 USDH
User Deposit Limit	5,000,000 USDH
Borrow limit	None

NFT Pool

SMBD

Parameter	Value
Optimal utilization rate (%)	80%
Loan to value ratio	20%
Liquidation penalty	10%
Liquidation threshold	50%
Min borrow APR	0%
Optimal borrow APR	50%
Max borrow APR	150%
Interest Rate Spread	20%
Borrow fee	2%
Flash loan fee	0.3%
Host fee percentage	20%
Deposit limit	1,500 SMBD
User Deposit Limit	100 SMBD
Borrow limit	None

USDC

Parameter	Value
Optimal utilization rate (%)	80%
Loan to value ratio	50%
Liquidation penalty	5%
Liquidation threshold	80%
Min borrow APR	0%
Optimal borrow APR	12%
Max borrow APR	150%
Interest Rate Spread	20%
Borrow fee	1%
Flash loan fee	0.3%
Host fee percentage	20%
Deposit limit	10,0000,000 USDC
User Deposit Limit	1,000,000 USDC
Borrow limit	None

SOL

Parameter	Value
Optimal utilization rate (%)	80%
Loan to value ratio	50%
Liquidation penalty	5%
Liquidation threshold	80%
Min borrow APR	0%
Optimal borrow APR	12%
Max borrow APR	150%
Interest Rate Spread	20%
Borrow fee	1%
Flash loan fee	0.3%
Host fee percentage	20%
Deposit limit	100,000 SOL
User Deposit Limit	10,000 SOL
Borrow limit	None

Coin98 Pool

Coin98

Parameter	Value
Optimal utilization rate (%)	80%
Loan to value ratio	65%
Liquidation penalty	10%
Liquidation threshold	75%
Min borrow APR	0%
Optimal borrow APR	12%
Max borrow APR	150%
Interest Rate Spread	15%
Borrow fee	1%
Flash loan fee	0.3%
Host fee percentage	20%
Deposit limit	1,000,000 C98
User Deposit Limit	100,000 C98
Borrow limit	None

UST

Parameter	Value
Optimal utilization rate (%)	80%
Loan to value ratio	0%
Liquidation penalty	5%
Liquidation threshold	85%
Min borrow APR	0%
Optimal borrow APR	12%
Max borrow APR	150%
Interest Rate Spread	15%
Borrow fee	0.5%
Flash loan fee	0.3%
Host fee percentage	20%
Deposit limit	0 UST
User Deposit Limit	0 UST
Borrow limit	None

UXD

Parameter	Value
Optimal utilization rate (%)	80%
Loan to value ratio	75%
Liquidation penalty	5%
Liquidation threshold	85%
Min borrow APR	0%
Optimal borrow APR	12%
Max borrow APR	150%
Interest Rate Spread	15%
Borrow fee	0.5%
Flash loan fee	0.3%
Host fee percentage	20%
Deposit limit	2,000,000 UXD
User Deposit Limit	200,000 UXD
Borrow limit	None

PAI

Parameter	Value
Optimal utilization rate (%)	80%
Loan to value ratio	75%
Liquidation penalty	5%
Liquidation threshold	85%
Min borrow APR	0%
Optimal borrow APR	12%
Max borrow APR	150%
Interest Rate Spread	15%
Borrow fee	0.5%
Flash loan fee	0.3%
Host fee percentage	20%
Deposit limit	2,000,000 PAI
User Deposit Limit	200,000 PAI
Borrow limit	None

USDH

Parameter	Value
Optimal utilization rate (%)	80%
Loan to value ratio	75%
Liquidation penalty	5%
Liquidation threshold	85%
Min borrow APR	0%
Optimal borrow APR	12%
Max borrow APR	150%
Interest Rate Spread	15%
Borrow fee	0.5%
Flash loan fee	0.3%
Host fee percentage	20%
Deposit limit	2,000,000 USDH
User Deposit Limit	200,000 USDH
Borrow limit	None

USDC

Parameter	Value
Optimal utilization rate (%)	80%
Loan to value ratio	75%
Liquidation penalty	5%
Liquidation threshold	85%
Min borrow APR	0%
Optimal borrow APR	12%
Max borrow APR	200%
Interest Rate Spread	15%
Borrow fee	0.5%
Flash loan fee	0.3%
Host fee percentage	20%
Deposit limit	4,000,000 USDC
User Deposit Limit	400,000 USDC
Borrow limit	None

UXD Pool

Pool Description/Risks

UXD Pool contains UXD, a decentralized stablecoin, backed 100% by positions. It also contains UXP, the governance token of UXD Protocol.

Users deposit SOL to mint UXD, and UXD hedges the price of SOL using perps. UXD is currently transitioning to the Asset Liability Module due to Mango (which UXD was initially based on being frozen post-exploit.

Read the detailed tokenomics of UXD/UXP here.

Oracles:

Both prices are detected using Switchboard, the links are available on the UI below the deposit button.

UXP

Parameter	Value
Optimal utilization rate (%)	80%
Loan to value ratio	50%
Liquidation penalty	20%
Liquidation threshold	65%
Min borrow APR	0%
Optimal borrow APR	30%
Max borrow APR	150%
Interest Rate Spread	20%
Borrow fee	1%
Flash loan fee	0.3%
Host fee percentage	20%
Deposit limit	30,000,000 UXP
User Deposit Limit	None
Borrow limit	None

UXD

Parameter	Value
Optimal utilization rate (%)	80%
Loan to value ratio	75%
Liquidation penalty	5%
Liquidation threshold	80%
Min borrow APR	0%
Optimal borrow APR	12%
Max borrow APR	150%
Interest Rate Spread	20%
Borrow fee	0.5%
Flash loan fee	0.3%
Host fee percentage	20%
Deposit limit	5,000,000 UXD
User Deposit Limit	None
Borrow limit	None

SOL

Parameter	Value
Optimal utilization rate (%)	80%
Loan to value ratio	75%
Liquidation penalty	5%
Liquidation threshold	80%
Min borrow APR	0%
Optimal borrow APR	12%
Max borrow APR	150%
Interest Rate Spread	20%
Borrow fee	0.5%
Flash loan fee	0.3%
Host fee percentage	20%
Deposit limit	250,000 SOL
User Deposit Limit	None
Borrow limit	None

USDC

Parameter	Value
Optimal utilization rate (%)	80%
Loan to value ratio	75%
Liquidation penalty	5%
Liquidation threshold	80%
Min borrow APR	0%
Optimal borrow APR	12%
Max borrow APR	200%
Interest Rate Spread	20%
Borrow fee	0.5%
Flash loan fee	0.3%
Host fee percentage	20%
Deposit limit	2,500,000 USDC
User Deposit Limit	None
Borrow limit	None

STEPN Pool

Pool Description/Risks

GMT/GST are tokens of the STEPN move-to-earn game. GMT is the governance token of the STEPN network, and GST is the utility token that users get rewarded with for running. GST can also be used for in-game utilities, such as minting new shoes or upgrading them.

Read the detailed tokenomics of GST/GMT here.

Max Utilization Risk

GST is frequently at 100% utilization, preventing liquidations & withdrawals. The protocol will manage this utilization by tweaking Max Borrow APR & Borrow Fees to discourage borrows at higher utilizations. Stay tuned to updates here.

At 100% utilization, users with GST deposits might not be able to withdraw. Users will have to wait for repays or deposits to occur before withdrawing.

Oracles:

The price of GMT/GST is detected using Switchboard oracles that pull data from Orca as well as multiple centralized exchanges (FTX/Binance/OKex & Gate). Currently, prices will be updated every 60s, or when a 0.5% move occurs.

Switchboard currently employs an equal weighting mechanism, where the average of these three prices are taken. In times of market congestion, we will bump the OracleBatchSize to 5 to improve oracle reliability.

Shadow Pool

Pool Description/Risks

SHDW is the governance token of the Shadow Protocol, run by developers behind GenesysGo (RPC provider) and associated with the Shadowy Super Coders NFT. Read more about them here.

Its tokenomics are dominated by 50% going to NFT holders via staking. Its liquidity is reliant on additional SHDW rewards distributed on Orca & Raydium. This liquidity mining program is financed from SHDW's strategic reserve in its tokenomics and has no estimated end date.

Some of the liquidity is provided via leverage yield farming (Tulip/Francium) which will cause liquidity to be more volatile after large price changes.

SHDW Token has minting enabled but no smart contract control for minting. There are no smart contract vulnerabilities for infinite minting or exploits to drain funds as of now.

The primary risk for SHDW is volatile price movement triggering liquidation for leveraged yield farmers, which reduces liquidity on chain for SHDW.

Oracles:

The price of SHDW is detected using a Switchboard oracle that pulls data from Orca's SHDW/SOL & SHDW/USDC pool, as well as the Serum Market ID. Currently, prices will be updated every 60s, or when a 0.5% move occurs.

Switchboard currently employs an equal weighting mechanism, where the average of these three prices are taken. In times of market congestion, we will bump the OracleBatchSize to 5 to improve oracle reliability.

Lending Pool

Pool Description/Risks

These are governance tokens of a variety of lending protocols on the Solana ecosystem.

They mostly have weak liquidity, with the exception of SLND with ~9MM of protocol owned liquidity. Parameters are set with this in mind, giving only 35% open LTV and 50% liquidation threshold to ensure that high slippage does not impact the overall health of the pool.

Each governance token has its associated risks such as smart contract risks of the underlying platform. (SLND -> Solend Platform smart contract risk).

Oracles:

The prices of these governance tokens are determined mainly using a Switchboard oracle that pulls data from on-chain sources, as well as centralized exchanges such as FTX or Ascendex. Currently, prices will be updated every 60s, or when a 0.5% move occurs.

Switchboard currently employs an equal weighting mechanism, where the average of these three prices are taken. In times of market congestion, we will bump the OracleBatchSize to 5 to improve oracle reliability.

Aurory Pool

Pool Description/Risks

AURY is the governance token of Aurory, a Solana-based blockchain gaming project. AURY can be used to purchase Nifties or used in-game as a utility token. Read more about their tokenomics here.

A possible risk with this pool is large unlock events causing volatile price changes, or slippages increasing as the price moves out of certain trading ranges. Do your own due diligence on AURY before investing in this pool.

Oracles:

The price of AURY is detected using a Switchboard oracle that pulls data from on-chain sources, and also centralized exchanges such as FTX and Huobi. Currently, prices will be updated every 60s, or when a 0.5% move occurs.

Switchboard currently employs an equal weighting mechanism, where the average of these three prices are taken. In times of market congestion, we will bump the OracleBatchSize to 5 to improve oracle reliability.

Basis Pool

Pool Description/Risks

BASIS is the utility token of basis.markets and can be staked into rBASIS to receive 16% of profits generated from the BASIS platform. This pool lets you use rBASIS as collateral, borrowing USDC or BASIS to add to your rBASIS deposit.

One risk of this pool is the smart contract risk associated with Basis's staking platform. Risk is low due to the simplicity of the contracts. When leveraging BASIS for rBASIS, make sure to manage your positions as utilization might drive up, turning the strategy unprofitable.

Oracles:

The price of BASIS is detected using a Switchboard oracle that pulls data from on-chain sources, Orca & Serum. rBASIS relies on the rBASIS:BASIS ratio calculated by drawing from the on-chain reserve of staked BASIS, and the number of rBASIS in circulation. Currently, prices will be updated every 60s, or when a 0.5% move occurs.

Switchboard currently employs an equal weighting mechanism, where the average of these three prices are taken. In times of market congestion, we will bump the OracleBatchSize to 5 to improve oracle reliability.

Kamino USDH Pool

Pool Description/Risks

USDH is an over-collateralized stablecoin created by Hubble Protocol. Hubble allows users to mint stablecoins with a minting fee against quality crypto assets such as SOL, BTC and ETH. The USDH Pool offers an alternative use-case for USDH, collateralizing short positions with USDH or borrowing USDH with 0% borrow fees but a borrow APY for short term borrows.

The biggest risk of the USDH pool is exploit or smart contracts risks related to Hubble or the USDH stablecoin. Hubble is audited 5 times by leading audit firms but always DYODD!

Oracles:

The price of USDH is detected using a switchboard oracle that pulls data from Saber's USDH/USDC LP exchange rate.

Switchboard currently employs an equal weighting mechanism, where the average of these three prices are taken. In times of market congestion, Solend will bump the OracleBatchSize to 5 to improve oracle reliability.

AMM Pool

Pool Description/Risks

AMM pool contains governance tokens of top Solana AMMs, such as Raydium or Orca. This pool lets you conduct pair trades, where you long A and short B to produce a market-neutral position that benefits from the overperformance of a protocol over another. AMM tokens are slightly more liquid than other governance tokens and have higher LTVs as such.

Oracles:

The price of these AMM tokens are predominately detected using Pyth, and Switchboard as a backup. For tokens without Pyth feeds, they rely on Switchboard ones.

Switchboard currently employs an equal weighting mechanism, where the average of various prices is taken. In times of market congestion, we will bump the OracleBatchSize to 5 to improve oracle reliability.

Nazare Stablecoin Pool

Pool Description/Risks

Nazare Stablecoin Pool contains the new Nazare LPs (nLPs) of USDC/USDT, UXD/USDC, and USH/USDC. This pool also contains stablecoins such as USH, UXD, USDC, and USDT, allowing users to leverage up on the Nazare LPs.

The LTVs of different stablecoins are set to 0; users will have to deposit LP tokens as collateral to borrow stablecoins and enjoy the incentives for doing so. This pool drives liquidity/TVL growth for Nazare, a new vault for the liquidity provision platform.

Nazare, as a new platform, is audited but represents a risk to the pool. Infinite minting bug or pricing errors for Nazare LPs is the main risk for the pool.

Oracles:

These LP tokens rely on a custom Switchboard feed to be priced based on Uniswap's V3 math. These oracles compute the price of the LP token by calculating the amount of each token in the LP, multiplied by the price of each stablecoin, divided by the number of LP Tokens in circulation.

As Switchboard prices are computed off-chain and published on-chain, these oracles are not susceptible to flash loan exploits.

Invictus Pool

What's Invictus Pool?

Invictus pool will now be deprecated due to the closure of the Invictus project.

Locked Staked IN (lsIN)

Parameter	Value
Optimal utilization rate (%)	80%
Loan to value ratio	40%
Liquidation penalty	15%
Liquidation threshold	55%
Min borrow APR	0%
Optimal borrow APR	30%
Max borrow APR	200%
Borrow fee	0.99%
Flash loan fee	0.3%
Host fee percentage	20%
Deposit limit	100,000 lsIN
User Deposit Limit	30,000 lsIN
Borrow limit	None

USDC

Parameter	Value
Optimal utilization rate (%)	80%
Loan to value ratio	75%
Liquidation penalty	5%
Liquidation threshold	80%
Min borrow APR	0%
Optimal borrow APR	30%
Max borrow APR	200%
Borrow fee	0.33%
Flash loan fee	0.3%
Host fee percentage	20%
Deposit limit	5,000,000 USDC
User Deposit Limit	500,000 USDC
Borrow limit	None

UST

Parameter	Value
Optimal utilization rate (%)	80%
Loan to value ratio	75%
Liquidation penalty	5%
Liquidation threshold	80%
Min borrow APR	0%
Optimal borrow APR	30%
Max borrow APR	200%
Borrow fee	0.33%
Flash loan fee	0.3%
Host fee percentage	20%
Deposit limit	5,000,000 UST
User Deposit Limit	500,000 UST
Borrow limit	None

Fees

Borrow APR

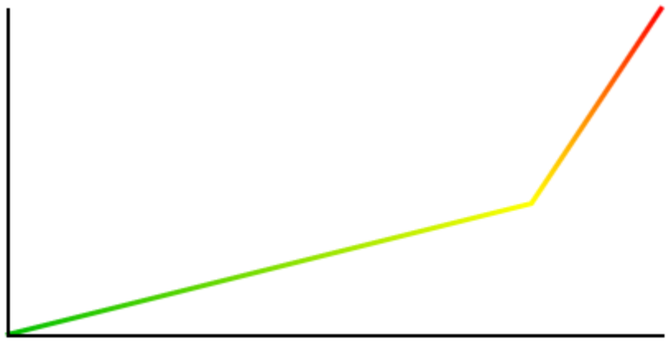
Solend charges an interest rate for borrowing assets. The borrow APR is set algorithmically, a function of utilization of the pool.

The following piecewise function is used:

$$\left\{x < r_{optimal} : \frac{x}{r_{optimal}} \cdot \left(b_{optimal} - b_{min}\right) + b_{min}\right\}$$

$$\left\{x \geq r_{optimal} : \frac{\left(x - r_{optimal}\right)}{1 - r_{optimal}} \cdot \left(b_{max} - b_{optimal}\right) + b_{optimal}\right\}$$

Which then produces the following graph:



Protocol Fees

Interest Rate Spread

Interest rate spread is a percentage of the borrow interest rate that Solend receives as protocol revenue. The interest spread is 20%.

Origination Fee

Origination fee is a fee charged upon the origination of a loan. Applied once when borrowed, upfront.

Solend's origination fees are broken down into a program fee and host fee. The host fee can be collected by any interface that refers loans to the protocol.

The fees for most vaults are set to 10 bips (0.1%). There's an 80/20 split between protocol and host fee. Check in-app or on the **Main Pool** a page for specific asset rates.

Liquidation Fee

Liquidation fee is a percentage of the liquidation penalty that Solend receives. These funds go to the DAO treasury. The liquidation fee is currently 30% of the liquidation penalty.

Solana Network Fees

There are two types of fees you pay on the Solana blockchain when transacting: transaction fees and rent fees.

Transaction fees

These fees compensate the validator network for the CPU/GPU and network resources necessary to process the state transaction. Transaction fees are extremely cheap (currently 0.000005 SOL).

Rent fees

These fees compensate the validator network for the memory resources that are required to store state in an account. Accounts which maintain a minimum balance equivalent to 2 years of rent payments are exempt from paying rent. When solend.fi is used, accounts are allocated on behalf of the user and funded with enough SOL to be rent-exempt. This amount can be recuperated later when the account is closed (though this isn't yet implemented on solend.fi). Rent fees can be expensive compared to transaction fees (~0.01 SOL), but are only paid the first time a user is interacting with Solend.

Liquidity Mining

LM 2.0

Starting July 1st, 2022, all SLND rewards on Solend will become options. Read the announcement here.

Details of the options:

- American style (can be exercised any time)
- Strike price of 30% of the price of SLND at the end of the period
- 1-year expiry
- Redeemable at the beginning of the next month as with other rewards

Exercise your SLND options by paying 10% of the total value in USDC and receive the 90% as your LM reward. APRs on the UI have accounted for the exercise cost.

Liquidity Mining

See announcement article here.

SLND is being emitted at a rate of 0.1585 per slot *(10M / slots per year* where slots are every 500ms or *10,000,000 / 63,072,000*). SLND rewards are distributed proportionally according to each market's weight.

For example, consider the sum of all weights is 28. If an asset's borrows are incentivized with 6x SLND, it means borrowers are rewarded 6/28 of the monthly SLND emissions.

Additionally, some assets are incentivized by partner rewards. The reward schedule is as follows:

Ongoing Rewards

These are more predictable and less likely to run out.

Asset	Supply	Borrow
SLND	2x SLND options	N/A
SOL	N/A	6x SLND options
USDC	N/A	9x SLND options
USDT	N/A	4x SLND options
stSOL	80k LDO + 1x SLND	N/A
mSOL	MNDE + 1x SLND	N/A

Period Rewards

Asset	Supply	Borrow	Start date	End date
N/A	N/A	N/A	N/A	N/A

There are currently no period rewards.

*Actual end date is defined by a slot number, which may diverge from the estimated date due to blocks processing faster or slower than expected.

Limits

Position Limits

An account can have at most 6 positions. A position is any supplied or borrowed asset. This limit is due to the Solana compute limit, since attempting to enter more positions would cause the transaction to fail. We plan to address this in the future.

For example, Alice is supplying {10 SOL} and is borrowing {1 SOL, 5 USDC}. She has 3 positions in total.

For now, the workaround for more positions is to use multiple accounts.

Audit

Solend has been audited by Kudelski. View the report here.

We have also used Soteria's scanner here, and cleared all vulnerabilities.

The Solend smart contracts are fully open source and have been since before the mainnet launch.

We built Solend using spl-token-lending as a starting point, which has been repeatedly audited/battle tested and used by many other teams in the space. We also intend to go through another round of audits if we make any major changes.

Bug Bounty

Program Overview

The bug bounty program covers the Solend smart contracts (no UI bugs) and is focused on preventing thefts and freezing of funds.

The Solend smart contracts are fully open source.

Rewards

Rewards are distributed according to the following classifications:

Severity	Max Prize
Critical	10% of value at risk, up to $1,000,000 USD
High	$50,000 USD
Medium	$5,000 USD

Severity is classified by the following:

Severity	Description
Critical	- Empty or freeze the contract's holdings (e.g. economic attacks, flash loans, reentrancy, MEV, logic errors, integer over-/under-flow) - Cryptographic flaws
High	- Token holders temporarily unable to transfer holdings - Users spoof each other - Theft of yield - Transient consensus failures
Medium	- Contract consumes unbounded gas - Block stuffing - Griefing denial of service (i.e. attacker spends as much in gas as damage to the contract) - Gas griefing

The actual prize amount is determined by a combination of factors including but not limited to severity, value at risk, and likelihood of being exploited.

Payouts are done in vesting SLND on Solana. This is anon-friendly (no KYC required).

Reporting

Email us a detailed description of the attack at security@solend.fi. Critical and High bug reports must come with a proof of concept.

Scope

Assets in Scope

Smart contract code

The main file of the lending program is here.

Impacts in Scope

Only the following impacts are accepted within this bug bounty program. All other impacts are not considered as in-scope, even if they affect the assets in the scope table.

Smart Contracts

- Loss of user funds staked (principal) by freezing or theft
- Loss of governance funds
- Theft of unclaimed yield
- Freezing of unclaimed yield
- Temporary freezing of funds for at least 1 hour
- Unable to call smart contract

Known Issues (not qualified)

Bug reports involving position limit, where a user can only have so many positions before actions fail due to the computation limit, are not accepted in this bug bounty program.

Bug reports involving borrow limit, where a user can borrow even when the limit is set, are not accepted in this bug bounty program.

Prioritized Vulnerabilities

We are especially interested in receiving and rewarding vulnerabilities of the following types:

Smart Contracts and Blockchain

- Re-entrancy
- Logic errors
  - Including user authentication errors
- Trust/dependency vulnerabilities
  - Composability vulnerabilities
- Oracle failure/manipulation
- Novel governance attacks
- Economic/financial attacks
  - Flash loan attacks
- Congestion and scalability
  - Running out of gas
  - Block stuffing
  - Susceptibility to front-running
- Consensus failures
- Cryptography problems
  - Signature malleability
  - Susceptibility to replay attacks
  - Weak randomness
  - Weak encryption
- Susceptibility to block timestamp manipulation
- Missing access controls / unprotected internal or debugging interfaces

Out of Scope Rules

The following vulnerabilities are excluded from the rewards for this bug bounty program:

- Attacks that the reporter has already exploited themselves, leading to damage
- Attacks requiring access to leaked keys/credentials
- Attacks requiring access to privileged addresses (governance, strategist)

Smart Contracts and Blockchain

- Incorrect data supplied by third party oracles
  - Not to exclude oracle manipulation/flash loan attacks
- Basic economic governance attacks (e.g. 51% attack)
- Lack of liquidity
- Best practice critiques
- Sybil attacks

The following activities are prohibited by this bug bounty program:

- Any testing with mainnet contracts; all testing should be done on devnet or private testnets
- Any testing with live pricing oracles or live third party smart contracts
- Attempting phishing or other social engineering attacks against our employees and/or customers
- Any testing with third party systems and applications (e.g. browser extensions) as well as websites (e.g. SSO providers, advertising networks)
- Any denial of service attacks
- Automated testing of services that generates significant amounts of traffic
- Public disclosure of an unpatched vulnerability in an embargoed bounty

Oracles

Solend's program primarily rely on Pyth and Switchboard as a backup.

These price feeds determine account health for liquidation.

Each oracle has feed-dependent refresh rates and can be checked out here:

Pyth & Switchboard

Introduction

Welcome to Solend!

Solend is a core DeFi primitive on Solana, that offers lending and borrowing on-chain. Integrating Solend improves capital efficiency by earning yields on idle funds, and empowers a large range of DeFi yield strategies to come to live.

Solend offers a kickback to integrations, allowing integrations to earn borrow fees or utilize our platform to generate higher yields or improve capital efficiency for your users.

Some ideas of integrations:

- DeFi Vaults (e.g. Yearn)
- GameFi game built on top of Solend mechanics
- Streaming yield-bearing cTokens for payroll (e.g. Sablier)
- Fixed yield protocol using cTokens
- Historical APY dashboard
- Margin Trading that borrows from Solend's reserves
- Account liquidation saver (repay loans for subscriber at reduced penalty before they get liquidated)
- Recursive deposit and borrow tool
- Prize-linked savings account (e.g. PoolTogether)

Solend is one of the easiest platforms to integrate on Solana, with our open sourced codebase here and our SDK here. We also have a lightweight client which demos usage of the SDK. We also have a dev portal here and our APIs here.

Feel free to get a “Scribe” role and reach out on Discord at our #dev-support channel, or DM Soju on Telegram @Sojuuuu54.

Devnet

The core team highly recommends testing on mainnet, on production! However, we maintain a devnet version of Solend. Devnet does not have as reliable oracles like mainnet, and to have all the latest features it's better to use production.

As native USDC is not on devnet as well, the actual token in our devnet reserves is just a token we minted. If you need it for testing, airdrop yourself SOL and deposit it into the program, to borrow "USDC" to test with.

The instructions is similar for both mainnet and devnet!

Alternatively, you can also fork Solend's code to test against. It is open-sourced here.

Risks

💡 Take note that Solend does not conduct due dilligence or verify the safety of each Permissionless Pool. Read this page to understand the risks of Permissionless Pools.

How does a user stay safe in Permissionless Pools?

1. Understand all the tokens present in any Permissionless Pool (even if you are depositing USDC).
2. Make sure the parameters are realistic.
3. Ensure that a liquidator is running in the pool.
4. Check that the oracle is displaying the correct price.

Note: Solend does not reimburse for losses incurred in a Permissionless Pool. It is important to conduct your own research on each pool.

Risks:

Let's assume a pool of the following two reserves when discussing risks:

SNDO: A governance token of a new project

USDC: Standard USDC

Token Risk

SNDO is a new token that you might not be familiar with. The risk of this token will affect the entire pool, even if you are just depositing USDC. For example, if a user managed to mint infinite numbers of SNDO, and deposits it to borrow USDC, it is possible to drain the pool. This could happen via an exploit by external parties or the team behind the token.

To assess this risk, you can research the mechanisms of the SNDO token, mainly around how new tokens enter the market (how tokens are minted), and the smart contract risks associated with the platform.

Liquidity Risk

As tokens can be listed permissionlessly with configs up to the creator, there is a chance of low-liquidity tokens being listed with "stable/safe" configs.

For example, if an open LTV of 75% is set, users can borrow $0.75 of USDC against $1 of SNDO. However, if SNDO is a new token with low liquidity, it might be common to have 15-20% slippage.

This means that a liquidator might not be able to reliably liquidate SNDO to pay back USDC debts, causing user's health to turn negative**,** and bad debt for the pool.

Liquidator Risk

For Permissionless Pools, Solend will not be operating a liquidator to ensure that all the pools are kept healthy. Pool creators are responsible for ensuring that a liquidator is operating for smaller tokens such as the SNDO token.

Users should use pools that have a reliable liquidator. Learn more about how to run one here.

Oracle Risk

For Permissionless Pools, pool creators are expected to create their own oracles. These oracles can be prone to error or used incorrectly (e.g. USDC oracle used for UST price). SNDO's price might be inaccurate or stale, causing liquidations or preventing users from withdrawing or borrowing.

We have a doc on Switchboard oracle listings here. Pool creators can follow it to learn how the existing oracles are created. Users can also read it to self-verify that an oracle is well created.

Pre-Listing Checklist

This guide is for users who want to create their own pool. It assumes familiarity with Solend and an understanding of basic DeFi principles. Permissionless Pools are powered by the same smart contracts that power Main and Isolated Pools.

Plan ahead for configurations

Start your pre-listing prep with this sheet (make a copy and fill it out). Plan ahead for the tokens you want to list along with their configurations. Their config definitions can be found here.

Eligible Tokens

In order to maintain the high quality UX on the Solend app, tokens must be on the token-list repo with a name, symbol, decimals and a high-def logo. If not, the token will not show up on the listing.

Reserves

For every token you want to list in the pool, you need one reserve with an initial deposit. To list a token, it must be on SPL-Token-List, and you need its mint address.

You will then need to deposit a small amount to start the pool. Depositing 0.01 X is enough.
Note: this deposit cannot be withdrawn.

Oracles

Assets in Main Pool are powered by both Pyth & Switchboard oracles, while most smaller cap tokens (SHDW, UXP, etc) only use Switchboard.

If the Solend team already maintains oracles for the token chosen, such as SOL/USDC/USDT they are automatically pre-filled for you in the permissionless pool creator UI.

If the token you are trying to list does not have an existing oracle, proceed to use Switchboard's Publisher + our guide to create your own oracle. This would be common for new tokens. Take note that there is a SOL fee for maintaining an oracle on-chain.

Configs

Next, you can set the configs you want for your pool. Configs such as LTVs, liquidation thresholds and APRs at different utilization. Fill in the numbers in this section if you have specific configs in mind.

Reference existing configs on Solend's dashboard.

If you are unsure of the configs, we have pre-filled option for Stable (stablecoins), Risky (for large cap tokens like SOL) or Very Risky in our pool creator UI later.

Liquidator

If you are listing a new token that isn't present on the Solend platform, you will need to set up a liquidator to maintain pool health. You can read about doing so here.

Creators are responsible for running a liquidator for their Permissionless Pool, our open sourced repo is here. If help is needed, feel free to come to the liquidator or dev-support channel in Discord.

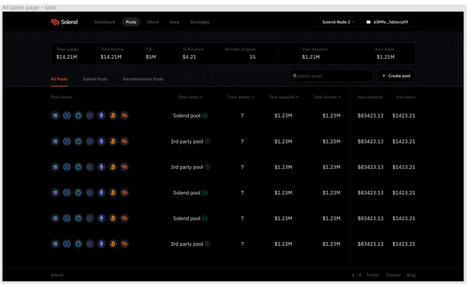
Final Checklist:

1. Do you clearly know what reserves you want to create and with what configurations?
2. Are your assets available on the token-list repo and on Jupiter?
3. Do you have your Pyth/Switchboard addresses for each reserve you want to create?
4. Is there a liquidator operating on all assets?
5. Is your Google sheet filled in with all the information?

Listing a Pool

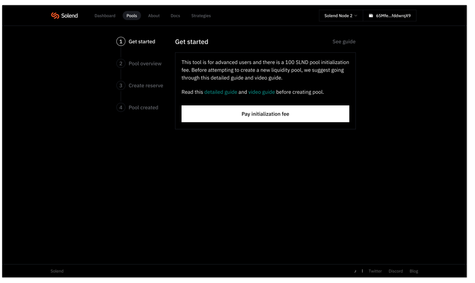
Now that you have a filled out the Google sheet, we can begin pool listing.

Proceed to Solend's All Pools page at: https://solend.fi/pools, and click on the "Create Pool" button on the top right corner.



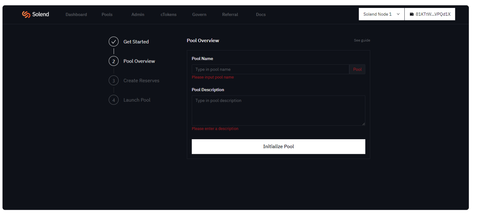
You will be directed to the Permissionless Pool creator.

Panel 1 - Get Started



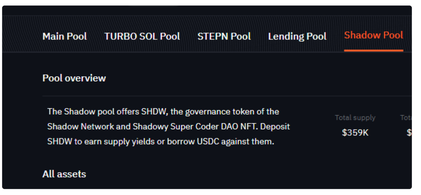
Creating your own pool requires an initialization fee of 200 SLND. You'll get 20% of the borrow fees generated by this pool in return!

Panel 2 - Pool Overview

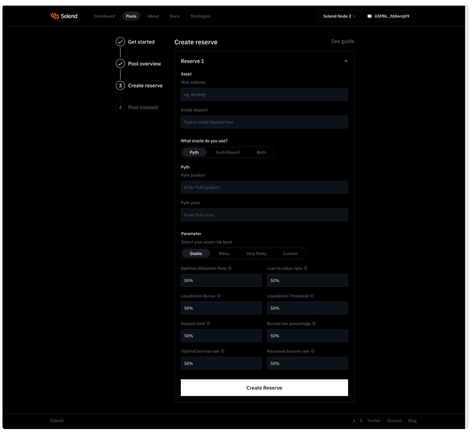


Next, type in the name of your pool and its description. Note that Solend automatically appends the word 'Pool' on the UI.

It will be displayed on Solend's UI for users of your pool:



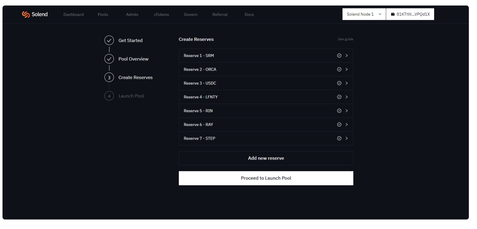
Panel 3 - Reserves Creation Page



You already have all of this information in your Google sheet! Simply copy and paste all the information over to these fields. Repeat until all the reserve details are keyed in.

Note that if you ever have to edit these details in the future, you will have to use our edit pool form here.

Panel 4 - Launch Pool

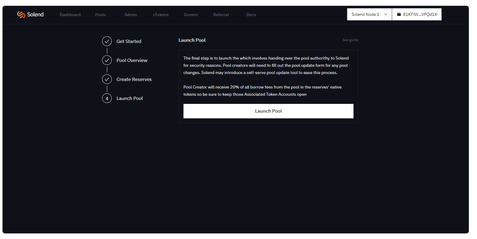


Hit "Register Pool" here and Solend will generate the pool creation transactions for you. Your wallet will pop up multiple times for you to approve certain transactions.

Make sure to do this step when the network is operating well as transactions might otherwise fail. Also note that you need around 0.2 SOL + some tokens of all the reserves you want to create.

After all the transactions go through, you can complete the pool creation process!

Panel 5 - Register Pool



This process hands authority over your pool over to Solend's smart contracts, and further edits will have to be done through the Edit Pool form here. This is the final step, and your pool will appear on our UI after 5 minutes. Make sure to keep your token accounts open to receive 20% of borrow fees generated!

Post-Listing

Now that you've created your pool, what's next?

1. Test the Pool
2. Bootstrap the Pool
3. Run Liquidator
4. Marketing to Users
5. Earn Borrow Fees

Test the Pool

Your pool is powered by the same contracts that power our Main & Isolated Pools and should work and feel the same.

Proceed to test the same actions (Deposit, Borrow, Repay or Withdraw) on your own pool.

Make sure your pool shows up correctly on the All Pools page and try it out by borrowing.

Bootstrap the Pool

When Solend launches a pool, it is usually bootstrapped with some small deposits & borrows to encourage activity and display a borrow APR. You can do the same by depositing and borrowing around $1,000~ in each reserve.

If you are listing a pool with a Governance Token + USDC, depositing some USDC for your users to borrow from is ideal.

Run Liquidator

After generating the lending market address, double check that the liquidator can liquidate this pool. Additionally, ensure that your liquidator is funded and operational.

Marketing to Users

Now that your pool is listed and bootstrapped, it's ready for users to use it. Copy & paste the Permissionless Pool link and announce it to your community and potential users!

Earn Borrow Fees

The pool creator is entitled to 20% of all the borrow fees generated by the platform, while the depositors earn the supply interest.

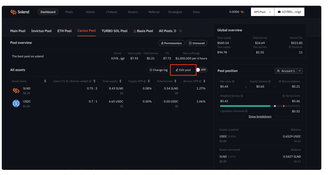
Be sure to keep your token accounts open to receive your share of the borrow fees!

Managing a Pool

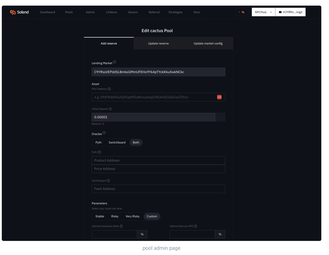
Permissionless pool creators have full ownership of their pools and can update parameters through the pool admin interface to add assets, update parameters, and update market properties

Pool admin page

If you are the owner of a permissionless pool, the "Edit pool" button will show. Clicking it opens the pool admin page.



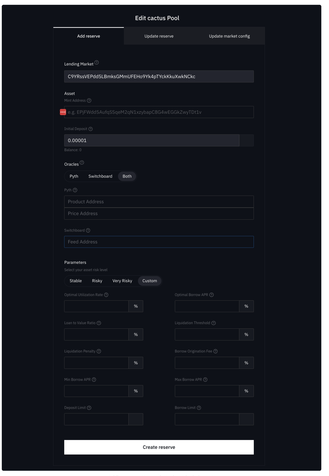
Screenshot of a permissionless pool. Note that the owner and connected wallet match.



pool admin page

Add asset

Pool owners may wish to list new assets. This can be achieved through the "Add reserve" panel on the admin admin page.



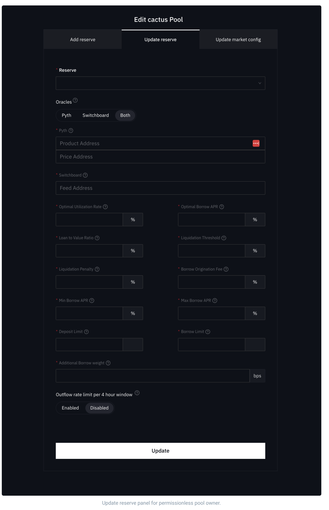
Adding a new reserve through pool admin page

Update existing assets

Asset parameters can be updated through the "Update reserve" panel. Be aware that updates to the reserve should be taken seriously as they may lead to user liquidations. Pool owners may wish to update a reserve's configuration to reflect the underlying token's risk, update supply/borrow apys, deprecate a token, set outflow limit, etc...

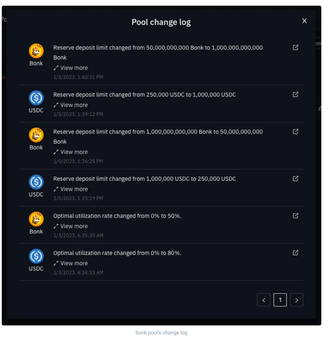
⚠️ **When in doubt, please reach out in the Solend Discord for assistance**

It is recommended for pool owners to give their users a 2 week notice before executing reserve updates so users may tweak their positions accordingly.



Update reserve panel for permissionless pool owner.

Changes to reserves can be observed through the blockchain and surfaced to users through the reserve changelog which is accessible on the pool's page



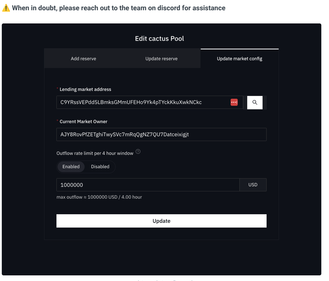
bonk pool's change log

Update market properties

A market contains one or more reserves and has control over its underlying reserves. Pool owners may use the update market config for the following reasons:

- **Transfer the market ownership to a different wallet or multisig**
  ⛔ Note: whoever holds the market owner, holds control over its reserves! If you accidentally send the market owner to the wrong address, nothing can be done to get it back so be extra careful when dealing with market property updates
- **Set global outflow limits for the market**. This param limits the damage possible due to an exploit. Using the screenshot below as an example, it means that a maximum of $10M can be borrowed or withdrawn from the Cactus pool in a 4 hour period

⚠️ **When in doubt, please reach out to the team on discord for assistance**



update market config panel

Switchboard v2 Guide

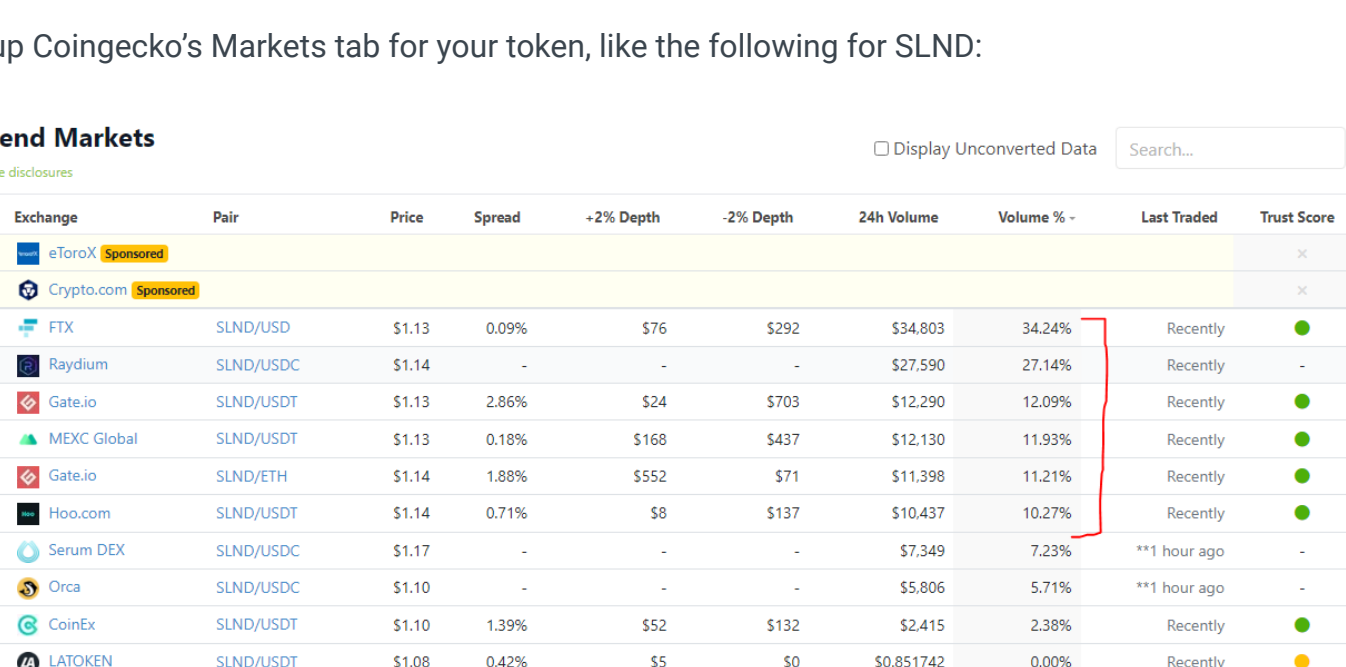
Creating a Switchboard oracle for Solend is easy by relying on their publisher here. Take note that creating oracles costs SOL, ~0.04 SOL a day will be consumed based on our configs.

This process might be a bit tough, so feel free to get help from Soju on Discord!

Deciding Where to Pull Prices from

Before we can start listing the oracle, we need to decide where the oracle will pull prices from. This would mainly be from centralized exchanges like FTX or Gate.io, or decentralized exchanges like Raydium or Orca.

Pull up Coingecko's Markets tab for your token, like the following for SLND:



Solend Markets

Affiliate disclosures

#	Exchange	Pair	Price	Spread	+2% Depth	-2% Depth	24h Volume	Volume %	Last Traded	Trust Score
*	eToroX Sponsored									
*	Crypto.com Sponsored									
1	FTX	SLND/USD	$1.13	0.09%	$76	$292	$34,803	34.24%	Recently	
8	Raydium	SLND/USDC	$1.14	-	-	-	$27,590	27.14%	Recently	-
4	Gate.io	SLND/USDT	$1.13	2.86%	$24	$703	$12,290	12.09%	Recently	
2	MEXC Global	SLND/USDT	$1.13	0.18%	$168	$437	$12,130	11.93%	Recently	
3	Gate.io	SLND/ETH	$1.14	1.88%	$552	$71	$11,398	11.21%	Recently	
5	Hoo.com	SLND/USDT	$1.14	0.71%	$8	$137	$10,437	10.27%	Recently	
9	Serum DEX	SLND/USDC	$1.17	-	-	-	$7,349	7.23%	**1 hour ago	-
10	Orca	SLND/USDC	$1.10	-	-	-	$5,806	5.71%	**1 hour ago	-
6	CoinEx	SLND/USDT	$1.10	1.39%	$52	$132	$2,415	2.38%	Recently	
7	LATOKEN	SLND/USDT	$1.08	0.42%	$5	$0	$0.851742	0.00%	Recently	

* Anomaly - Trading price is an outlier against the average

We can see that most of the trading volume for SLND takes place on FTX, Gate and MXC for centralized exchanges, and Raydium for decentralized exchanges.

Let's pull prices from these sites.

Fetching Prices from Web2

To pull prices from centralized exchanges, we will use pre-built scripts for each exchange listed on the Appendix of this page. Alternatively, you can do a Web2 fetch under the Switchboard UI (centralized exchanges are Web2).

Let's run through the steps for FTX SLND/USD. Click 'Import' and copy and paste the FTX script from the Appendix. Take note that the API link is "https://ftx.com/api/markets/gst/usd", this would give you the price of GST instead. Replace "GST" with "SLND", "https://ftx.com/api/markets/slnd/usd". FTX quotes prices in terms of USD, so there is no need to convert from USDT and USDC to USD.

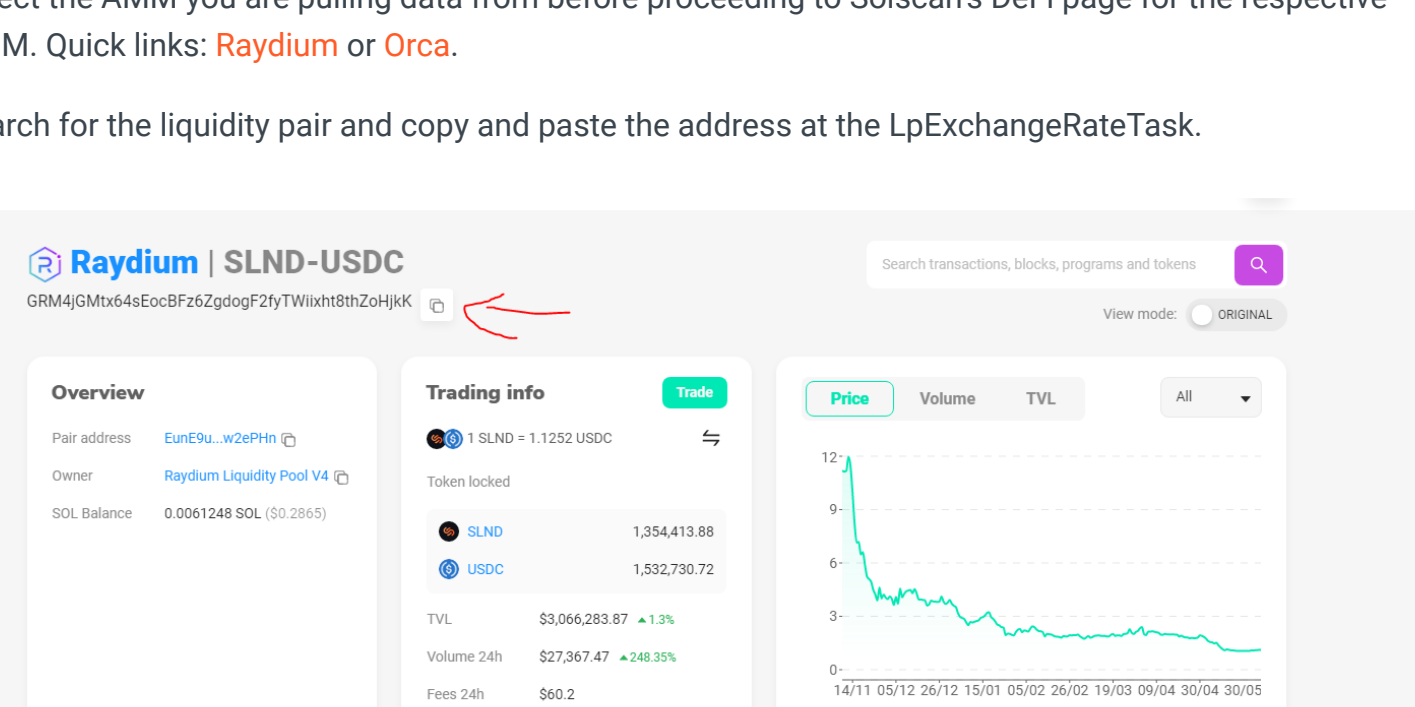
For Gate, we need to add an extra step for the "multiplyTask". Gate quotes SLND in terms of USDT, so we need to convert USDT to USD. We do this via the second half of the script, where it says:

```
    {
      "multiplyTask": {
        "aggregatorPubkey": "ETAaeeuQBwsh9mC2gCov9WdhJENZuffRMXY2HgjCcSL9"
      }
```

In the Appendix, we have the oracle aggregators for USDC/USD, USDT/USD, and SOL/USD. Since Gate quotes in terms of USDT, we need to add in the USDT/USD oracle to convert the USDT price to the USD price. Simply make sure that the "aggregatorPubkey: <USDT/USD Oracle>".

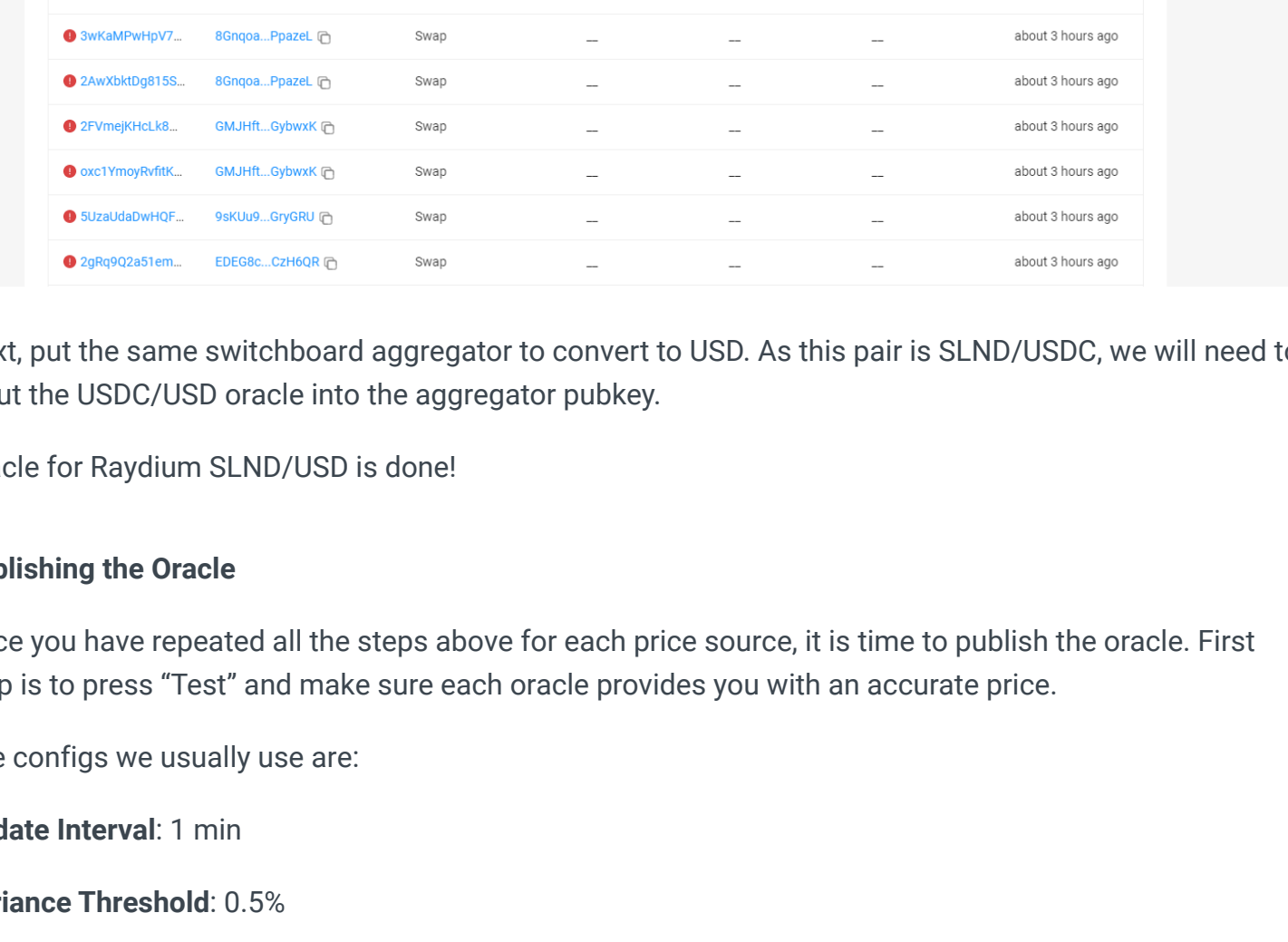
Fetching Prices from Web3

For Orca or Raydium LPs, we will need to fetch the "LpExchangeRate" of a liquidity pair, such as SLND/USDC, and do the same multiply task as above.



Select the AMM you are pulling data from before proceeding to Solscan's DeFi page for the respective AMM. Quick links: Raydium or Orca.

Search for the liquidity pair and copy and paste the address at the LpExchangeRateTask.



Next, put the same switchboard aggregator to convert to USD. As this pair is SLND/USDC, we will need to input the USDC/USD oracle into the aggregator pubkey.

Oracle for Raydium SLND/USD is done!

Publishing the Oracle

Once you have repeated all the steps above for each price source, it is time to publish the oracle. First step is to press "Test" and make sure each oracle provides you with an accurate price.

The configs we usually use are:

Update Interval: 1 min

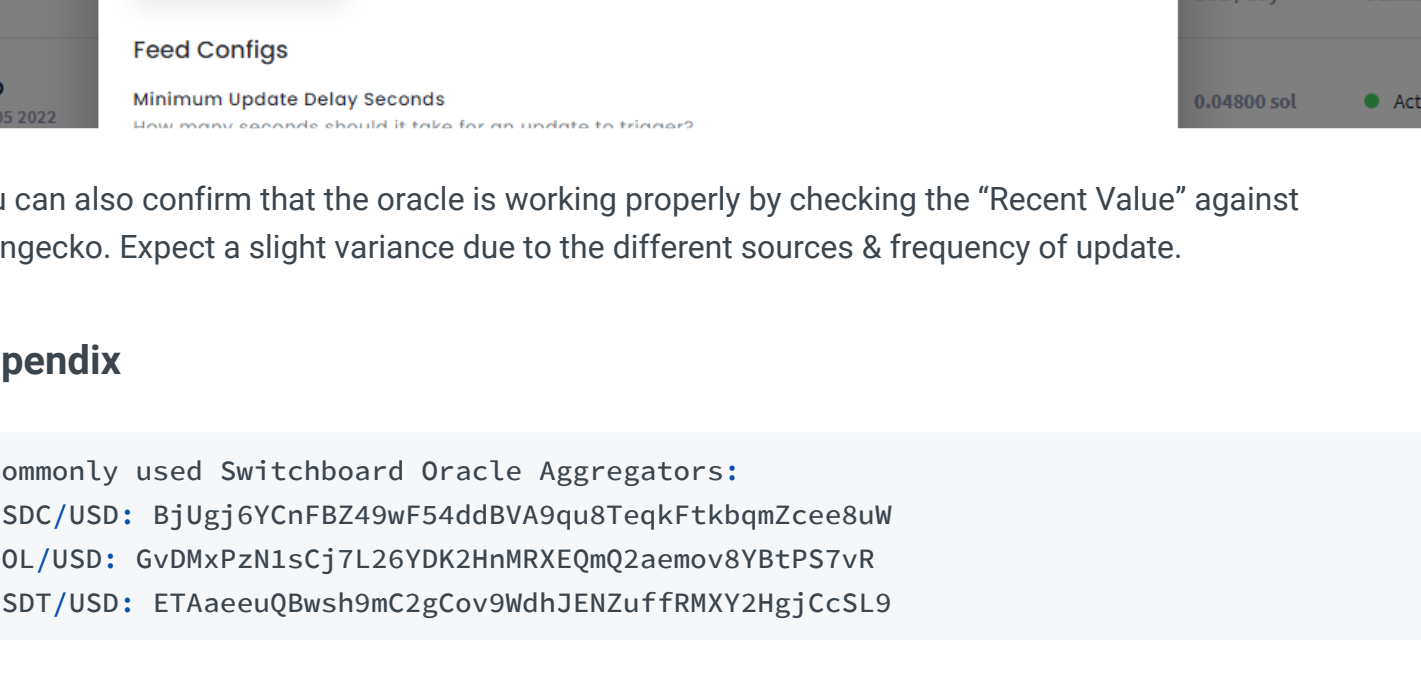
Variance Threshold: 0.5%

Force Report Period: 1 min

This means that your oracle will be updated every minute, or when a price move of 0.5% occurs and has the same configs as what the core team uses.

Getting the Oracle ID

The final step is to copy and paste the oracle ID where you need to use it.



You can also confirm that the oracle is working properly by checking the "Recent Value" against Coingecko. Expect a slight variance due to the different sources & frequency of update.

Appendix

```
Commonly used Switchboard Oracle Aggregators:
USDC/USD: BjUgj6YCnFBZ49wF54ddBVA9qu8TeqkFtkbqmZcee8uW
SOL/USD: GvDMxPzN1sCj7L26YDK2HnMRXEQmQ2aemov8YBtPS7vR
USDT/USD: ETAaeeuQBwsh9mC2gCov9WdhJENZuffRMXY2HgjCcSL9
```

Huobi:

```
{
  "name": "Huobi GST/USD",
  "tasks": [
    {
      "httpTask": {
        "url": "https://api.huobi.pro/market/detail/merged?symbol=gstusdt"
      }
    },
    {
      "medianTask": {
        "tasks": [
          {
            "jsonParseTask": {
              "path": "$.tick.bid[0]"
            }
          },
          {
            "jsonParseTask": {
              "path": "$.tick.ask[0]"
            }
          }
        ]
      }
    },
    {
      "multiplyTask": {
        "aggregatorPubkey": "ETAaeeuQBwsh9mC2gCov9WdhJENZuffRMXY2HgjCcSL9"
      }
    }
  ]
}
```

FTX:

```
{
  "name": "FTX GST/USD",
  "tasks": [
    {
      "httpTask": {
        "url": "https://ftx.com/api/markets/gst/usd"
      }
    },
    {
      "jsonParseTask": {
        "path": "$.result.price"
      }
    }
  ]
}
```

Binance:

```
{
  "name": "BinanceCom BTC/USD",
  "tasks": [
    {
      "httpTask": {
        "url": "https://www.binance.com/api/v3/ticker/price?symbol=BTCUSDT"
      }
    },
    {
      "jsonParseTask": {
        "path": "$.price"
      }
    },
    {
      "multiplyTask": {
        "aggregatorPubkey": "ETAaeeuQBwsh9mC2gCov9WdhJENZuffRMXY2HgjCcSL9"
      }
    }
  ]
}
```

MEXC:

```
{
  "name": "Mxc BTC/USD",
  "tasks": [
    {
      "httpTask": {
        "url": "https://www.mexc.com/open/api/v2/market/ticker?symbol=LARIX_USDT"
      }
    },
    {
      "medianTask": {
        "tasks": [
          {
            "jsonParseTask": {
              "path": "$.data[0].ask"
            }
          },
          {
            "jsonParseTask": {
              "path": "$.data[0].bid"
            }
          },
          {
            "jsonParseTask": {
              "path": "$.data[0].last"
            }
          }
        ]
      }
    },
          {
            "multiplyTask": {
              "aggregatorPubkey": "ETAaeeuQBwsh9mC2gCov9WdhJENZuffRMXY2HgjCcSL9"
            }
          }
        ]
      }
```

AscendEX:

```
{
  "tasks": [
    {
      "httpTask": {
        "url": "https://ascendex.com/api/pro/v1/spot/ticker?symbol=JET/USDT"
      }
    },
    {
      "medianTask": {
        "tasks": [
          {
            "jsonParseTask": {
              "path": "$.data.close"
            }
          },
          {
            "jsonParseTask": {
              "path": "$.data.ask[0]"
            }
          },
          {
            "jsonParseTask": {
              "path": "$.data.bid[0]"
            }
          },
          {
            "multiplyTask": {
              "aggregatorPubkey": "ETAaeeuQBwsh9mC2gCov9WdhJENZuffRMXY2HgjCcSL9"
            }
          }
        ]
      }
    }
  ]
}
```

Bitfinex:

```
{
  "name": "Bitfinex BTC/USD",
  "tasks": [
    {
      "httpTask": {
        "url": "https://api-pub.bitfinex.com/v2/tickers?symbols=tOXYUST"
      }
    },
    {
      "medianTask": {
        "tasks": [
          {
            "jsonParseTask": {
              "path": "$[0][1]"
            }
          },
          {
            "jsonParseTask": {
              "path": "$[0][3]"
            }
          },
          {
            "jsonParseTask": {
              "path": "$[0][7]"
            }
          },
          {
            "multiplyTask": {
              "aggregatorPubkey": "ETAaeeuQBwsh9mC2gCov9WdhJENZuffRMXY2HgjCcSL9"
            }
          }
        ]
      }
    }
  ]
}
```

Pool Ideas

List any tokens

With permissionless pools, anyone can come to Solend and list their new token in an isolated pool! Users can set parameters based on how new the token is, and get them increased over time as the liquidity and holders grow.

This is the first/original use case for permissionless pools and makes our goal of "Any SPL token as collateral" possible.

Experiment with different pool ideas

Permissionless pools also allow DeFi degens to create different pools based on certain strategies.

For example, a mSOL / SOL pool with 98% LTV might be interesting. This would allow for higher leverage in the mSOL/SOL strategy, where you borrow SOL <6% APY and stake it with Marinade for 6-7% APY.

Uncollateralized loans for protocols

Protocols can also set up a permissionless pool that allows them to borrow without collateral from. This allows protocols like Delta One to borrow funds and conduct leveraged defi strategies for example.

Protocols can do this by creating a pool with two reserves, USDC and another token they have mint authority over. They can then set the price of this token to a flat $1, and then mint themselves collateral accordingly.

To minimize bad debt or default, this pool should be governed by the integrated protocols' smart contracts to manage position sizes and margin. Funds should not enter anyone's wallets at all times. If interested, reach out to Soju on Discord!

Uncollateralized loans for individuals/companies

Alternatively, a user can create a permissionless pool to borrow money as an individual or company such as a market making firm.

Similar to how protocols do it, but instead of a protocol minting a token a user does it instead. This would allow the pool creator to borrow uncollateralized from users who deposit into the pool. DYODD is a must when depositing in these pools.

This would have to rely on depositors trusting this borrower/pool creator and is a higher risk compared to relying on the integrated smart contracts.

Building Blocks

Lending Market Owner/Authority:

This is the account that owns the entire pool -there is one for Main, Isolated, and etc.

Lending Market (State)

This contains all the reserves.

Reserves State (Market-level Positions)

For every pool, each token has its own reserve. This is where supply/balances are tracked, and is updated using the account value * pyth_price pulled from oracles.

Obligations State (User-level positions)

Each user's position is called an obligation, and is updated using RefreshObligation by calling Pyth_price + the % of Liquidity provided (for supply) or cumulative borrow rate (for borrows)

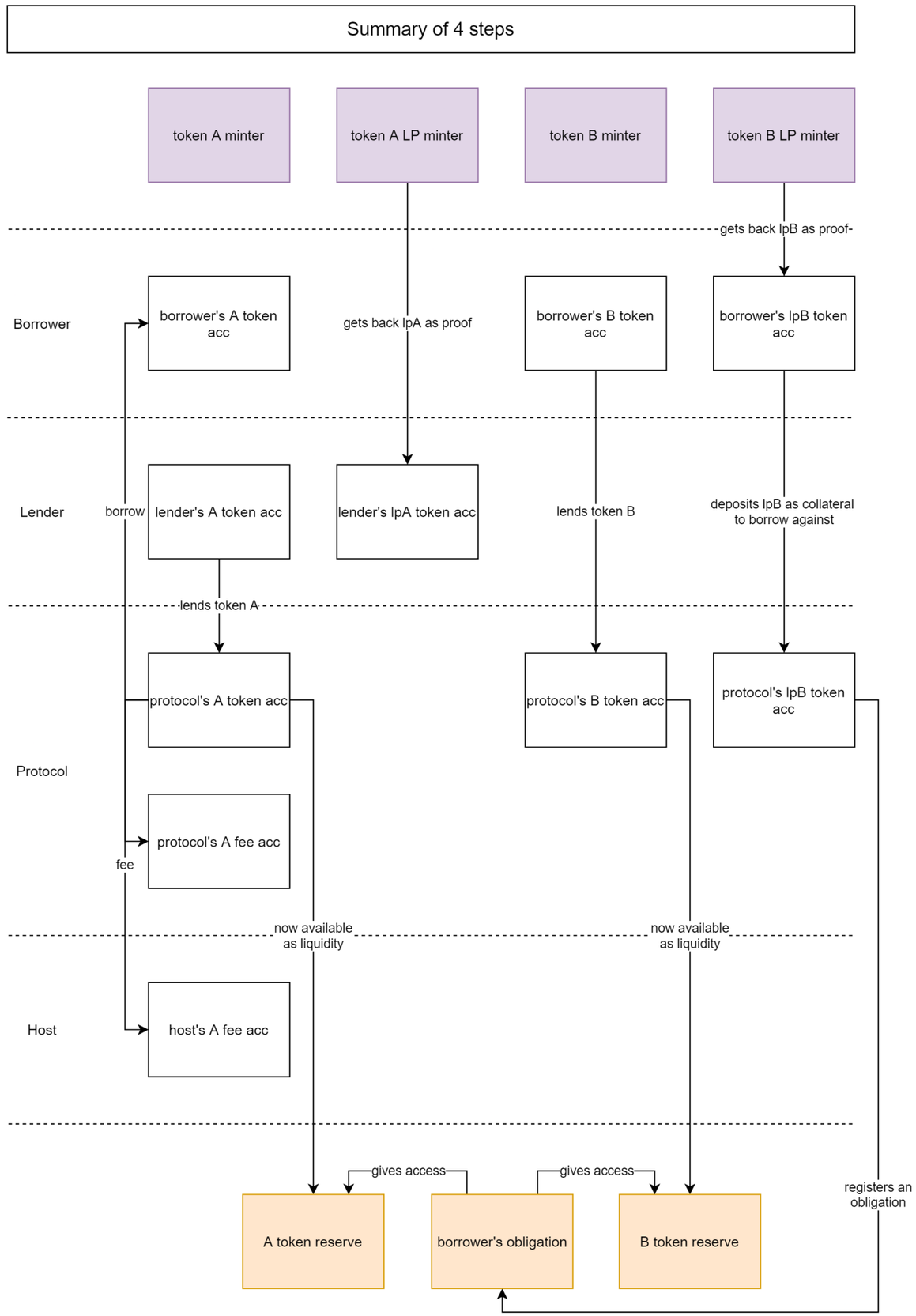
Reserves State:

- Asset: Reserve Token
- Address: Reserve Address that contains the ATA of the liquidity
- CollateralMintAddress: cToken Mint Address
- CollateralSupplyAddress: Where cTokens are held to be used as collateral
- Liquidity Address: ATA of the Reserve that contains the tokens
- LiquidityFeeReceiverAddress: Where borrow fees generated are held before being swept into the protocol treasury

Example:


```
      },
      {
        "asset": "UST",
        "address": "Ab48bKsiEzdm481mGaNVmv9m9DmXsWWxcYHM588M59Yd",
        "collateralMintAddress": "nZtL8HKX3aQtk3VpdvtdwPpXF2uMia522Pnan2meqdf",
        "collateralSupplyAddress": "4HXDioboWL85gQocYNkWM62AB5ctrf8jVykSVco67Lzx",
        "liquidityAddress": "5LyHdTXh1MSbRzE7xfTtfpV8W5eaySJnSiTs6FdHhrSo",
        "liquidityFeeReceiverAddress": "4GctGML68E1kDcvskGTXRPY9ngxmxVnJXjpsJ68YBXPR",
        "userSupplyCap": 5000000
      },
      {
```

Software Flowchart



Take a look at the flowchart overview of Solend protocol. Credit to ilmoi on GitHub here.

Access Controls

Our smart contracts are all open sourced, located here.

Description	Address
Program	`So1endDq2YkqhipRh3WViPa8hdiSpxWy6z3Z6tMCpAo`
Upgrade authority	`2Fwvr3MKhHhqakgjjEWcpWZZabbRCetHjukHi1zfKxjk`

Our Solend Program, "`So1endDq2YkqhipRh3WViPa8hdiSpxWy6z3Z6tMCpAo`", is owned by the BPF Upgradeable Loader program which lets the Upgrade Authority "`GDmSxpPzLkfxxr6dHLNRnCoYVGzvgc41tozkrr4pHTjB`" upload a new program to make changes/improvements. We hold the keypair of the Upgrade Authority, so we utilize that to push upgrades to mainnet.

For now, there isn't a timelock program, but transitioning to governance in the future will cause upgrades to go through a governance vote as well.

However, minor changes to mainnet program are reviewed internally across multiple members of the team, or by external engineers from the Solana team. We will likely conduct another audit if we perform major changes to the codebase.

Our code is not "Anchor verified" yet, but we will look into doing so soon.

Description	Address
Lending market owner	`5pHk2TmnqQzRF9L6egy5FfiyBgS7G9cMZ5RFaJAvghzw`
Fee receiver	`9RuqAN42PTUi9ya59k9suGATrkqzvb9gk2QABJtQzGP5`

This lending market owner contains all the reserves for our main/isolated pools. When updating the configs such as reserve limit or parameters, we pass a tx through the lending market owner such as this one to update the configs. This can only change the configs and can't move funds from the user.

An example of configs we can change is setting deposit/borrow limit to 0. We do this when deprecating or delisting certain assets. However, we can't set withdrawal to 0 and lock up user funds.

User Instructions

Interact program to program via these instructions, or refer to the GitHub link below:

solana-program-library/instruction.rs at mainnet · solendprotocol/solana-program-library
GitHub

DepositReserveLiquidityAndObligationCollateral (Supply)

- Deposit reserve liquidity (Transfer) + mint cTokens (MintTo) + Transfer cTokens into CollateralSupplyAddress (Transfer)
- Does not require RefreshReserve as it is impossible to worsen an obligation's health by repaying.
- Sample TX

WithdrawObligationCollateralAndRedeemReserveCollateral

- Redeem cTokens from CollateralSupplyAddress
- Exchange cTokens into Tokens using the cToken Rate
- Requires RefreshReserve to ensure that the user is able to withdraw without defaulting their existing borrows
- RefreshReserve → RefreshObligation → WithdrawObligationCollateralAndRedeemReserveCollateral

BorrowObligationLiquidity

- Bundled with Refresh Reserve
- if "borrow_reserve.last_update.is_stale(clock.slot)? {" fails, reserve is considered stale (refreshreserve + borrow must be in the same slot)
- RefreshReserve → RefreshObligation → Borrow

RepayObligationLiquidity

- Does not require RefreshReserve as it is impossible to worsen an obligation's health by repaying.

cToken Instructions:

DepositReserveLiquidity

- Deposit funds into the reserve, mint cToken
- No need to escrow the cTokens in the CollateralSupplyAddress
- Relies on cToken:Token Ratio

RedeemReserveCollateral

- Withdraw Funds, burn cTokens
- Relies on cToken:Token Ratio

Monitoring/Misc Instructions

RefreshReserve

- Update the price of every reserve, usually done before RefreshObligation
- Pull Oracle prices
- Update CumulativeBorrowRate to calculate interest paid since last refreshed block
- Update the Obligation CumulativeBorrowRate to apply an increase in borrow value
- Update cToken Rate for supply side users
- Done 2x for every reserve, supply and borrow side

RefreshObligation

- Done after RefreshReserves
- Calculate individual obligations that ran the Withdraw or Borrow tx to calculate LTV before borrows
- Assess whether the tx can go through based on account health

Notes:

RefreshReserve/RefreshObligation used to be required for all transactions, but were removed to reduce oracle reliance after Jan 2022's network issues.

Liquidator will call RefreshObligation before liquidating

However, borrow/withdraw has to check whether the obligation is refreshed in the SAME slot. But obligations require that reserves are refreshed.

RefreshReserve → RefreshObligation (repeated through all the positions) requires a lot of memory, which is limited by Solana, and thus limits our positions to 6.

Computing Supply & Borrows

Monitoring Instructions:

- RefreshReserve - handles all the tracking work and updates ratio
- RefreshObligation - computes obligation balances based on reserve ratios

For Supply:

- cToken Ratio acts as a % ownership of the LP
- For a pool with 99 USDC, if you deposit 1 USDC, you will own 1% of the pool
- When the RefreshReserve occurs, the pool now has 110 USDC. Your 1% is now worth 1.1 USDC.
- Total Reserves / cTokens Minted

For Borrows:

- APY is used to update CumulativeBorrowRate as an index
- When borrowing, calculate the initial CumulativeBorrowRate (let's say 1)
- RefreshReserve → Updates the CumulativeBorrowRate (let's say its 1.02)
- Obligation borrows would be increased by *1.02/1 or 2%

```
/// Compound current borrow rate over elapsed slots
    fn compound_interest(
        &mut self,
        current_borrow_rate: Rate,
        slots_elapsed: u64,
    ) -> ProgramResult {
        let slot_interest_rate = current_borrow_rate.try_div(SLOTS_PER_YEAR)?;
        let compounded_interest_rate = Rate::one()
            .try_add(slot_interest_rate)?
            .try_pow(slots_elapsed)?;
        self.cumulative_borrow_rate_wads = self
            .cumulative_borrow_rate_wads
            .try_mul(compounded_interest_rate)?;
        self.borrowed_amount_wads = self
            .borrowed_amount_wads
            .try_mul(compounded_interest_rate)?;
        Ok(())
    }
}
```

Why: For efficiency, the RefreshReserve just has to update the cToken Rate or the CumulativeBorrowRate instead of updating every obligation’s supply or borrow numbers. Tracking is only done at the ratio/index level, and never at an individual level unless changes to the user's positions are being made.

cTokens

https://solend.fi/ctokens

cTokens are a *yield-bearing deposit receipt*. It means you can convert USDC into cUSDC to get a tradeable token that also earns interest on Solend. You can hold this token in your wallet to continue earning interest, or you can send it to someone else, and they'll earn interest.

cTokens open up a world of possibilities for developer integrations. Because cTokens are just regular SPL tokens, they're easily composable and compatible with just about everything! If you're working on something that uses Solend cTokens, check out our Developer Portal and Grants Program!

Take note that cTokens currently do not have liquidity mining, such as additional SLND or MNDE rewards. You can mint cTokens here.

solana-program-library/processor.rs at 400dd876c51862c21122f7d4378a7c82d996b0ef · s…
GitHub

Normally, if you are using our Main Lend/Borrow, your cTokens will be held in custody by our smart contracts to be used as "collateral" for borrowing against it. This tool is mainly for developers to integrate our cTokens into their protocols, and then direct their users here to mint their cTokens.

cToken Ratio Calculations:

- cToken Ratio acts as a % ownership of the LP
- For a pool with 99 USDC, if you deposit 1 USDC, you will own 1% of the pool
- When the RefreshReserve occurs, the pool now has 110 USDC. Your 1% is now worth 1.1 USDC.

cToken Ratio is calculated by taking the Total USDC in the reserves / cUSDC in Circulation.

cToken Addresses

cToken Address / CollateralMintAddress

These are the cToken addresses that you can find using Solscan, and are also known as 'CollateralMintAddress' in our code.

Main Pool

Description	Address
cSOL	`5h6ssFpeDeRbzsEHDbTQNH7nVGgsKrZydxdSTnLm6QdV`
cUSDC	`993dVFL2uXWYeoXuEBFXR4BijeXdTv4s6BzsCjJZuwqk`
csoETH	`AppJPZka33cu4DyUenFe9Dc1ZmZ3oQju6mBn9k37bNAa`
cBTC	`Gqu3TFmJXfnfSX84kqbZ5u9JjSBVoesaHjfTsaPjRSnZ`
cSRM	`4CxGuD2NMr6zM8f18gr6kRhgd748pnmkAhkY1YJtkup1`
cUSDT	`BTsbZDV7aCMRJ3VNy9ygV4Q2UeEo9GpR8D6VvmMZzNr8`
csoFTT	`A38TjtcYrfutXT6nfRxhqwoGiXyzwJsGPmekoZYYmfgP`
cORCA	`E9LAZYxBVhJr9Cdfi9Tn4GSiJHDWSZDsew5tfgJja6Cu`
cRAY	`2d95ZC8L5XP6xCnaKx8D5U5eX6rKbboBBAwuBLxaFmmJ`
cSBR	`4GAGuewTRMfBJukgu3HSzpT48iqP4bYVCq3tygnjqmxL`
cMER	`BsWLxf6hRJnyytKR52kKBiz7qU7BB3SH77mrBxNnYU1G`
cmSOL	`3JFC4cB56Er45nWVe29Bhnn5GnwQzSmHVf6eUq9ac91h`
cETH	`FbKvdbx5h6F86h1pZuEqv7FxwmsVhJ88cDuSqHvLm6Xf`
cSLND	`D3Cu5urZJhkKyNZQQq2ne6xSfzbXLU4RrywVErMA2vf8`
cscnSOL	`AFq1sSdevxfqWGcmcz7XpPbfjHevcJY7baZf9RkyrzoR`
cstSOL	`QQ6WK86aUCBvNPkGeYBKikk15sUg6aMUEi5PTL6eB4i`
cUST	`nZtL8HKX3aQtk3VpdvtdwPpXF2uMia522Pnan2meqdf`
cFTT	`DiMx1n2dJmxqFtENRPhYWsqi8Mhg2p39MpTzsm6phzMP`
cUSDT-USDC	`6XrbsKScacEwpEW5DVNko9t5vW3cim9wktAeT9mmiYHS`
cmSOL-SOL	`4icXEpFVMrcqob6fnd3jZ6KjKrc6cqre6do1f8kKAC1u`

Isolated Pools

Description	Address
cSOL (TURBO SOL)	`AVxnqyCameKsKTCGVKeyJMA7vjHnxJit6afC8AM9MdMj`
cUSDC (TURBO SOL)	`HKijBKC2zKcV2BXA9CuNemmWUhTuFkPLLgvQBP7zrQjL`
clsIN (Invictus)	`5Jk2vER2x9WGAZmVGEafBtkEzjYB4mcvvVHbxMcbprhJ`
cUSDC (Invictus)	`DR1b9VzSe8o658UVJL9b6XzqkRmurcmLujpZRaBL3yDU`
cUST (Invictus)	`5vxqpDHcHhhJBwNdgeFjfJ814qL36S4fJneSihYccZNg`
cxSTEP (Step)	`E1hgwtGqjT4po2vg1LFGvf5XiZgUZBQeabcstQympPPa`
cSTEP (Step)	`9XtoqcLnc1psQuTPTGjfEwwabzBeUeFihvyZptcALwph`
cSOL (Step)	`7criSZai4hqKukScLUa5W2UV6Q9pjFMdJjQphKavBqur`
cUSDC (Step)	`7tBwN1tWTNsepwJcuD46zPBu6FuivWMvN2J5zP6HoPYg`
cFIDA (FIDA)	`ArohPh341JrQsWk7paiUwJKDe4MjvhtE5XxjYUoiM2fH`
cSOL (FIDA)	`2fHFudEbottFSbpe8oX5KyCDrnx8tfgL8PFj67iz9Y2h`
cUSDC (FIDA)	`7ziVqwra9BTyzXwFTL6JBMY4LtHdQZeQ9cqLkTKeYvoq`
cATLAS (ATLAS)	`EHFzasjYT1VmeKtvPtPCytJugQFBoYdhXXBt2WazinfY`
cPOLIS (ATLAS)	`ZeTubq2tcT9s8sPGKihum5scLGoQDqtYd6BYAy15y2P`
cUSDC (ATLAS)	`HwdCShcytebyeLaU79FB5cD4RDJVWnN7vs5kCJr6nKf7`
cSAMO (SAMO)	`E9qQ9pdZjk9zvm7Qp3UUynddJMadK85F8ZNqWyeUHcjV`
cUSDC (SAMO)	`9zCPUDEF5oGymtihmPQL6bUUxarno7w6qzM7Amiyh9bw`
cUSDC (Stable)	`4JZ6PXqRDp8jQxXUYX9cbAzHi6uzZk856aoAqPGdV5Da`
cUSDT (Stable)	`9wCxum1oB9JpgPUuVmsPTqDDvKeQqP3rnGFg3GEMrUGT`
cPAI (Stable)	`DyJZX45rgh9nADa19tsFeSV69ZwvX4UqPoPP39mdBiDq`
cUST (Stable)	`5czXDWSVFjH16hJGvwZTE8sMbh4vQggZ1gMhyVE4RWgx`
cUXD (Stable)	`3R3mzc8o9oXCsBX2dKG7Bzc3ov1m7t4UHtb81ktAeCxY`
cUSDH (Stable)	`DE9WN39kGuqZwsBpmd8Fs8F7b4T38nzpiNo8DseznLBU`
cSMBD (NFT)	`5NBAYi6ELmNCvxy1QQ3iLaNRidRSBM9R8DN4bVK6kXM2`
cUSDC (NFT)	`Awi5FDTwSH5XKH11kDLynccJPQbvKgkGFaekExSRk5QM`
cSOL (NFT)	`83xjxt6wQzci6zgEukhswGVwesHC94D6C7q6fUZaAThb`
cC98 (C98)	`VTyR5PvnbNAp7uAg7kr9cfAhNZEodoJrCTRGC67pAjP`
cUST (C98)	`DWYPVVz79kAbbiL42rWtmWoyvDX9M7aGLyf9Cu7Ewvds`
cUXD (C98)	`ErJswCkk3oRS9poFdRxJHt6j9yQisTB8YQAqJkE7iC5U`
cPAI (C98)	`CKUk55h1JcyK7DsvSYjYVw5XiEgrB2LgXMfyuBBnRmHT`
cUSDH (C98)	`AGhAmBTQGHhDWSwmBCL91GRyv7FP3HRZarmRK6XccnM7`

Addresses

Going forward, use our endpoint to grab addresses.

We will be deprecating https://github.com/solendprotocol/common soon.

SLND Token

`SLNDpmoWTVADgEdndyvWzroNL7zSi1dF9PC3xHGtPwp`

Solend Contracts

Description	Address
Program	`So1endDq2YkqhipRh3WViPa8hdiSpxWy6z3Z6tMCpAo`
Upgrade authority	`2Fwvr3MKhHhqakgjjEWcpWZZabbRCetHjukHi1zfKxjk`
Lending market owner	`5pHk2TmnqQzRF9L6egy5FfiyBgS7G9cMZ5RFaJAvghzw`
Fee receiver	`9RuqAN42PTUi9ya59k9suGATrkqzvb9gk2QABJtQzGP5`

Mainnet

Glossary

Description	Address
Lending market	Pubkey of the Authority over all the reserves
Vault	Where all the underlying tokens are held (= TVL)
pythProgramID	Pyth's Oracle program ID
switchboardProgramID	Switchboard's Oracle program ID

Token X

Description	Address
Reserve	Account address of the Reserve
CollateralMintAddress	Token address of the cTokens
CollateralSupplyAddress	Where cTokens are held/escrowed to be borrowed against
LiquidityAddress	Where the Tokens are held
LiquidityFeeReceiver Address	Where we collect fees generated from borrowing

Main Pools

Main Pool

Description	Address
Lending market	`4UpD2fh7xH3VP9QQaXtsS1YY3bxzWhtfpks7FatyKvdY`
Vault	`DdZR6zRFiUt4S5mg7AV1uKB2z1f1WzcNYCaTEEWPAuby`
pythProgramID	`FsJ3A3u2vn5cTVofAjvy6y5kwABJAqYWpe4975bi2epH`
switchboardProgramID	`DtmE9D2CSB4L5D6A15mraeEjrGMm6auWVzgaD8hK2tZM`

Reserve

Description	Address
SOL	`8PbodeaosQP19SjYFx855UMqWxH2HynZLdBXmsrbac36`
USDC	`BgxfHJDzm44T7XG68MYKx7YisTjZu73tVovyZSjJMpmw`
soETH	`3PArRsZQ6SLkr1WERZWyC6AqsajtALMq4C66ZMYz4dKQ`
BTC	`GYzjMCXTDue12eUGKKWAqtF5jcBYNmewr6Db6LaguEaX`
SRM	`5suXmvdbKQ98VonxGCXqViuWRu8k4zgZRxndYKsH2fJg`
USDT	`8K9WC8xoh2rtQNY7iEGXtPvfbDCi563SdWhCAhuMP2xE`
soFTT	`2dC4V23zJxuv521iYQj8c471jrxYLNQFaGS6YPwtTHMd`
RAY	`9n2exoMQwMTzfw6NFoFFujxYPndWVLtKREJePssrKb36`
SBR	`Hthrt4Lab21Yz1Dx9Q4sFW4WVihdBUTtWRQBjPsYHCor`
MER	`5Sb6wDpweg6mtYksPJ2pfGbSyikrhR8Ut8GszcULQ83A`
mSOL	`CCpirWrgNuBVLdkP2haxLTbD6XqEgaYuVXixbbpxUB6`
ETH	`CPDiKagfozERtJ33p7HHhEfJERjvfk1VAjMXAFLrvrKP`
SLND	`CviGNzD2C9ZCMmjDt5DKCce5cLV4Emrcm3NFvwudBFKA`
scnSOL	`DUExYJG5sc1SQdMMdq6LdUYW9ULXbo2fFFTbedywgjNN`
stSOL	`5sjkv6HD8wycocJ4tC4U36HHbvgcXYqcyiPRUkncnwWs`
UST	`Ab48bKsiEzdm481mGaNVmv9m9DmXsWWxcYHM588M59Yd`
FTT	`8bDyV3N7ctLKoaSVqUoEwUzw6msS2F65yyNPgAVUisKm`
USDT-USDC LP	`9mZsd1b9cN7JyqJvkbqhVuTfg8PAuKjuhPxpcsVNjYoC`
mSOL-SOL LP	`6ve8XyELbecPdbzSTsyhYKiWr7wg3JpjfxE1cqoN9qhN`

Isolated Pools

TURBO SOL

Description	Address
Lending Market	7RCz8wb6WXxUhAigok9ttgrVgDFFFbibcirECzWSBauM
Vault	55YceCDfyvdcPPozDiMeNp9TpwmL1hdoTEFw5BMNWbpf

Reserves

Description	Address
SOL	UTABCRXirrbpCNDogCoqEECtM3V44jXGCsK23ZepV3Z
USDC	EjUgEaPpKMg2nqex9obb46gZQ6Ar9mWSdVKbw9A6PyXA

cTokens

cSOL	AVxnqyCameKsKTCGVKeyJMA7vjHnxJit6afC8AM9MdMj
cUSDC	HKijBKC2zKcV2BXA9CuNemmWUhTuFkPLLgvQBP7zrQjL

Invictus Pool

Description	Address
Lending Market	5i8SzwX2LjpGUxLZRJ8EiYohpuKgW2FYDFhVjhGj66P1
Vault	6N6tqnemGoR5pUdtKKp3FvdD94Gi98f2ySEo1dzZ2Uqv

Reserves

Description	Address
IsIN	4XYbgZJirfnwjmpJKQgMQEvjncYFi2CsPTFzBguCjCjG
USDC	AuT5vA4bScsaJBiyNHnAttKToTCHj4Kwi4sg8bCyPPr8
UST	7MymBKwTPPMC4A9Ktwc1F2V5Xw7Kj3DqvRYUvLk2SF4h

cTokens

Description	Address
cIsIN	5Jk2vER2x9WGAZmVGEafBtkEzjYB4mcvvVHbxMcbprhJ
cUSDC	DR1b9VzSe8o658UVJL9b6XzqkRmurcmLujpZRaBL3yDU
cUST	5vxqpDHcHhhJBwNdgeFjfJ814qL36S4fJneSihYccZNg

Step Pool

Description	Address
Lending Market	DxdnNmdWHcW6RGTYiD5ms5f7LNZBaA7Kd1nMfASnzwdY
Vault	csotR9rcbLV3bCzBKxNJ3GjYhzH9cXffZX3TAQpw4oG

Reserves

Description	Address
xSTEP	HH9Aig5MAvMNcivGfAbWU5Da9nfiTwBaYJBK2KZyZppn
STEP	C5ozcRb4PJeJvakPeGgm9bgwcL6rPcKPfV95d2owW86C
SOL	7trBAMkVU8dcPQVdScz7VNywZwqnD1rwXkwkVPQJ95bT
USDC	FCU2wpx3ED1dY7bKszzcyxUVNTduLurUEmCGGv2w3Lfm

cTokens

Description	Address
cxSTEP	E1hgwtGqjT4po2vg1LFGvf5XiZgUZBQeabcstQympPPa
cSTEP	9XtoqcLnc1psQuTPTGjfEwwabzBeUeFihvyZptcALwph
cSOL	7criSZai4hqKukScLUa5W2UV6Q9pjFMdJjQphKavBqur
cUSDC	7tBwN1tWTNsepwJcuD46zPBu6FuivWMvN2J5zP6HoPYg

Bonfida Pool

Description	Address
Lending Market	91taAt3bocVZwcChVgZTTaQYt2WpBVE3M9PkWekFQx4J
Vault	76Asux4XZYqrP61G52eRZRQ6GCUPQUmYme3hTCaNgmxv

Reserves

Description	Address
FIDA	A3ZhKMuwHygRqjXiMDqM2PyeT35Z1LiDUqwrtjiHn89M
SOL	3WPYWiZtc2uJq1JiF3Z3KswicFAp5VrFgEHwP3CkuDUn
USDC	EBRtjgHJiEBYnQ5QzTGcBxTwbapEQ3bvh1BrgaGzhX9e

cTokens

Description	Address
cFIDA	ArohPh341JrQsWk7paiUwJKDe4MjvhtE5XxjYUoiM2fH
cSOL	2fHFudEbottFSbpe8oX5KyCDrnx8tfgL8PFj67iz9Y2h
cUSDC	7ziVqwra9BTyzXwFTL6JBMY4LtHdQZeQ9cqLkTKeYvoq

Star Atlas Pool

Description	Address
Lending Market	99S4iReDsyxKDViKdXQKWDcB6C3waDmfPWWyb5HAbcZF
Vault	2vhoVMQWEc12dUEEcJ6w8j3ZnrA4Tk6w8pPFhoWfVsUy

Reserves

Description	Address
ATLAS	8RX5oDxnydPPsA92epWnyXrrM26w7JgAQoVVt9kbiZwq
POLIS	29Znf6g5qmRfTdnbyRQUWvMt94Gzn2KPCzY2ixxY9Mnt
USDC	5hVVs474TRejwwfqsecNp97riQGDtSmhTV6jiWSxJfWR

cTokens

Description	Address
cATLAS	EHFzasjYT1VmeKtvPtPCytJugQFBoYdhXXBt2WazinfY
cPOLIS	ZeTubq2tcT9s8sPGKihum5scLGoQDqtYd6BYAy15y2P
cUSDC	HwdCShcytebyeLaU79FB5cD4RDJVWnN7vs5kCJr6nKf7

Dog Pool

Description	Address
Lending Market	HASr6hiYVoRcVXk3GttC4PjBBPQ3sGYDzE7HSPJdcke6
Vault	DLokPHis2W5f3jQ4Kgv5PQHebkun2KQtsXcFidhR19Za

Reserves

Description	Address
SAMO	HwZSKqyo2QQ2YCzF282ZrpH4JRQWf3Qad1fWHtKCDZjx
USDC	5VuBkDYXcV1svRm1BKShA2wqKsszYWjPwoT4Q32YXcp3

cTokens

Description	Address
cSAMO	E9qQ9pdZjk9zvm7Qp3UUynddJMadK85F8ZNqWyeUHcjV
cUSDC	9zCPUDEF5oGymtihmPQL6bUUxarno7w6qzM7Amiyh9bw

Stable Pool

Description	Address
Lending Market	GktVYgkstojYd8nVXGXKJHi7SstvgZ6pkQqQhUPD7y7Q
Vault	Ej4KxxUz73edQzjfsPVWvYxT5eyhQoWoXpo7BYm2Ejhj

Reserves

Description	Address
USDC	JCRDg9T5mUUxazdJ2nGWDN2pdcXVQc5VM8XDp1DW6Aoa
USDT	aMyCaGf7MwsqpgoeioVGXZytadsjFZ6euuYSadLcXbY
PAI	Bd5pFCJf9AafuXZkfkU6SDZVpeE5xAqX69bs8nDSxd2J
UST	A57FVdDgcyz1NserCSMSWaWDyfWZw77ikfXvE2cwPF18
UXD	27YJsVpHWvjS8BKaz7Gd8unSFJAMrh6gPEFjqhYxn9AE
USDH	NoTf6a9khWa5cCh6v5RRronH7YuatY7gDWmdKUPoBhM

cTokens

Description	Address
cUSDC	4JZ6PXqRDp8jQxXUYX9cbAzHi6uzZk856aoAqPGdV5Da
cUSDT	9wCxum1oB9JpgPUuVmsPTqDDvKeQqP3rnGFg3GEMrUGT
cPAI	DyJZX45rgh9nADa19tsFeSV69ZwvX4UqPoPP39mdBiDq
cUST	5czXDWSVFjH16hJGvwZTE8sMbh4vQggZ1gMhyVE4RWgx
cUXD	3R3mzc8o9oXCsBX2dKG7Bzc3ov1m7t4UHtb81ktAeCxY
cUSDH	DE9WN39kGuqZwsBpmd8Fs8F7b4T38nzpiNo8DseznLBU

NFT Pool

Description	Address
Lending Market	29yTiqjGdoNiRLMVc7ZoqFpbW3gkmefwMG9SUiMMD4J9
Vault	BtDeXoN6L5s5Nodvny3qvwK55b3YLcFkL7KqvKD7bFNz

Reserves

Description	Address
SMBD	EpZipddM7X7TofTzypVaz9GkjBziWgZZCgPz9HhGAB1t
USDC	8WpwDA1qs28vrkS3q1B8pfEXNP5MMkAc9qhiQZeYBK9N
SOL	8xogd14bBxBdGKDfkDciPPp6pZ3Cw4Yj5USRbGJDbZpA

cTokens

Description	Address
cSMBD	5NBAYi6ELmNCvxy1QQ3iLaNRidRSBM9R8DN4bVK6kXM2
cUSDC	Awi5FDTwSH5XKH11kDLynccJPQbvKgkGFaekExSRk5QM
cSOL	83xjxt6wQzci6zgEukhswGVwesHC94D6C7q6fUZaAThb

Coin98 Pool

Description	Address
Lending Market	7tiNvRHSjYDfc6usrWnSNPyuN68xQfKs1ZG2oqtR5F46
Vault	8web9hJK4TQJBV23WQpBw9jMvn3YE1EV3PEcnXJvgwQa

Reserves

Description	Address
Coin98	9bVRrxPjXBxM6rEyTLcR2opvdA2UGhdDwL8CLLm1b8KP
UST	B5513y5wt161CxLX5U2o5cGFHYbcGMuTKc6yu1ni3AbC
UXD	46Lh1P2XmTNG8Gnt4zkTdG1BXi2V18NggfYTbXpSzAYy
PAI	A3cQwWXzsaC5nfLDf7cbakZeBAJFGf1qMxvnf4yDRUUJ
USDH	3eDDvVgyxZ7aWLjLmKDYpeGHidCH7jkfHXcCXtpGqNKg

cTokens

Description	Address
cC98	VTyR5PvnbNAp7uAg7kr9cfAhNZEodoJrCTRGC67pAjP
cUST	DWYPVVz79kAbbiL42rWtmWoyvDX9M7aGLyf9Cu7Ewvds
cUXD	ErJswCkk3oRS9poFdRxJHt6j9yQisTB8YQAqJkE7iC5U
cPAI	CKUk55h1JcyK7DsvSYjYVw5XiEgrB2LgXMfyuBBnRmHT
cUSDH	AGhAmBTQGHhDWSwmBCL91GRyv7FP3HRZarmRK6XccnM7

UXD Pool

Description	Address
Lending Market	HCuEqcXaGeioiJf5vNMTyQC7HMPqJm5aZPkSjA2qceDS
Vault	AwT1ixXdQcjQMgDWpzyNQhMyuU3z2hbMGcEYT9sHBAsU

Reserves

Description	Address
UXP	BvZwsK3v8gqHqhNMDnGsD7z2V83bGzw82o2vrqU3kvBC
UXD	GseYnT313pSLAVidDwKWzo48QRFMmaoF2QLk3fK8DpZU
SOL	3x4swTGLgBkCDmxXYQRP96J1evCgy3ZsnUf9YJzVPtNy
USDC	Br6ucRrPNrJXPMa31FGBqS9WH86YGMQvu8Lv4dma8R5k

cTokens

Description	Address
UXP	Gqi3AojEduDqgyNuKzTwW3QPwUD55J2WzbCuYaqCDMiU
UXD	GDkx8oKUf5rHx1UNCaMfvjwEGa8uZy6ECYmzGh3Pkxef
SOL	29icHQwfhSgSo3MYjtAgCXKyfFn6RETz2sREEtgfaXy4
USDC	5vsr61rdgfeCtWTZHv54y7jkJ4xKg9WQDT4UusBsnePf

STEPN Pool

Description	Address
Lending Market	BjsAGLZzAgBUsiaTTDQv7PWDUDL9dQfKvYwb4q6FoDuD
Vault	CqEyTyxQDSwqFcrp7GXqEhhyeDNGvYrhyDkWtUcaa9oG

Reserves

Description	Address
GMT	BqYvhari2P9mXYLqcgJv41kRUvLhBfGoQXGWHnw26UJw
GST	CaacWq72hHubpwi92UUVsoE4xDNKqr1MhRwYeqH3YPAU
USDC	DxdbG2YwFFJeePbNFZRXuF2Sx34B1NTxWNbeEHdo4rzb

cToken

Description	Address
GMT	C8Pm2aMMdvHHBV1QDFzTGMrTs5k6MdDjxBwJEXX3aJ5U
GST	GHuZTLtRqFH6s6nkRBaaHFrVuVpepuvHsNApX4zvP3Hg
USDC	9E4YoiXauf22frwmioB7WFkQ2XAaTgKV7J7YsvM4TAN

Shadow Pool

Description	Address
Lending Market	Foo9vqN6fj1NyymHmD1gwZkgVgEqzNSrwyeqyoLYGe7j
Vault	79X5SSdcWcs9cFFByhSX2TBnoYmaU7BkJ6ajrpJPwxqe

Reserves

Description	Address
SHDW	27wbnZxrA8Gq7UMYaRFJPP4ijr6HuWZZNXAovEHrG6Pn
USDC	4JvgJDxBmXrrYzdLAN4sAbyGYVWVw6mGjJwnaHFVTNWY

cToken

Description	Address
SHDW	Bk2fPSoxvvm6bS1C5M6HSMrgofLCQmjoFDM9StTXJo4R
USDC	A7CsM2A83J1z7hGec45MGsRyqSLULdcLtxvrvPvvUoo6

Devnet

Devnet

Description	Address
Program	`ALend7Ketfx5bxh6ghsCDXAoDrhvEmsXT3cynB6aPLgx`
Lending Market	`GvjoVKNjBvQcFaSKUW1gTE7DxhSpjHbE69umVR5nPuQp`
pythProgramID	`CqFJLrT4rSpA46RQkVYWn8tdBDuQ7p7RXcp6Um76oaph`
switchboardProgramID	`7azgmy1pFXHikv36q1zZASvFq5vFa39TT9NweVugKKTU`

Reserves

Description	Address
SOL	`5VVLD7BQp8y3bTgyF5ezm1ResyMTR3PhYsT4iHFU8Sxz`
USDC	`FNNkz4RCQezSSS71rW2tvqZH1LCkTzaiG7Nd1LeA5x5y`
ETH	`CuQcLfN3iTWqyEEbAHNZp7awChbR4KQPFJsTUhsAxrcu`
BTC	`FNXBRv3saDgVoaoDLV5ho28D4AvZikG9r7QpThTLak9f`
SRM	`5YKmMZcuWEdF5NavNK4MgKPRhnhw6jq22zwcdh1dvxbd`
USDT	`ERm3jhg8J94hxr7KmhkRvnuYbKZgNFEL4hXzBMeb1rQ8`
soFTT	`8eUYeEiGXRMN7V9Xx7CehwSRfJB9iQfJW249zX1z7Uue`
RAY	`FpKhrtU6nDjEcAi3SsSevRRgPzirLkZpmXD6nfeb7k8c`
SBR	`5CXDzn4AygQwu59fGxo34T965BmYMHuGvAuoRMeRBKg4`
MER	`HbXi7Gpe5GXzE7V1SjEEy3sDurmFi7RbTvCZ3Rd7Nv6b`
mSOL	`Ei2dC7hFxBhVq5pn4qNJLMxKBSojx7p7wnNoBz4HELKG`

Introduction

Permissionless Pools allow anyone to create an isolated pool on Solend. The creator decides and earns 20% of origination fees generated in that pool. Therefore, a successful Permissionless Pool means fees generated for the pool creator.

Permissionless Pools unlock the following:

- **Ability to list tokens faster**: The core Solend team can only list so many tokens. Allowing anyone to list Permissionless Pools themselves means opening the floodgates for anyone to list any asset.
- **Utility for any token:** The idea that any token can be listed is a powerful one. With this, any asset wrapped as an SPL token has utility in being accepted as collateral for a loan.
- **Protocol owned market**: With 20% of origination fees generated in the pool going to the creator, user growth of the pool rewards both Solend and the pool creator. Protocols and influencers can create their own pools and promote it to their audience. Similar to what Uniswap did for tokens in DeFi Summer, Permissionless Pools unlock utility for the long tail of assets.

If you have any questions, ask us in the #permissionless-pools channel on Discord!

Oracles

Solend relies on oracles to retrieve the price of a token for liquidation or account health calculations.

Creators can create Switchboard V2 oracle fees relatively easily by following this guide. Permissionless Pools will be powered by Pyth and Switchboard oracles.

All Pools Page

With this launch, a UI revamp is coming on our “All Pools” page! You can easily see which pools are permissionless and which ones are created by the Solend team. As Permissionless Pools have higher levels of risk and due diligence needed.

You can also click “Create Pool” to create your own pool via this UI.

Liquidators

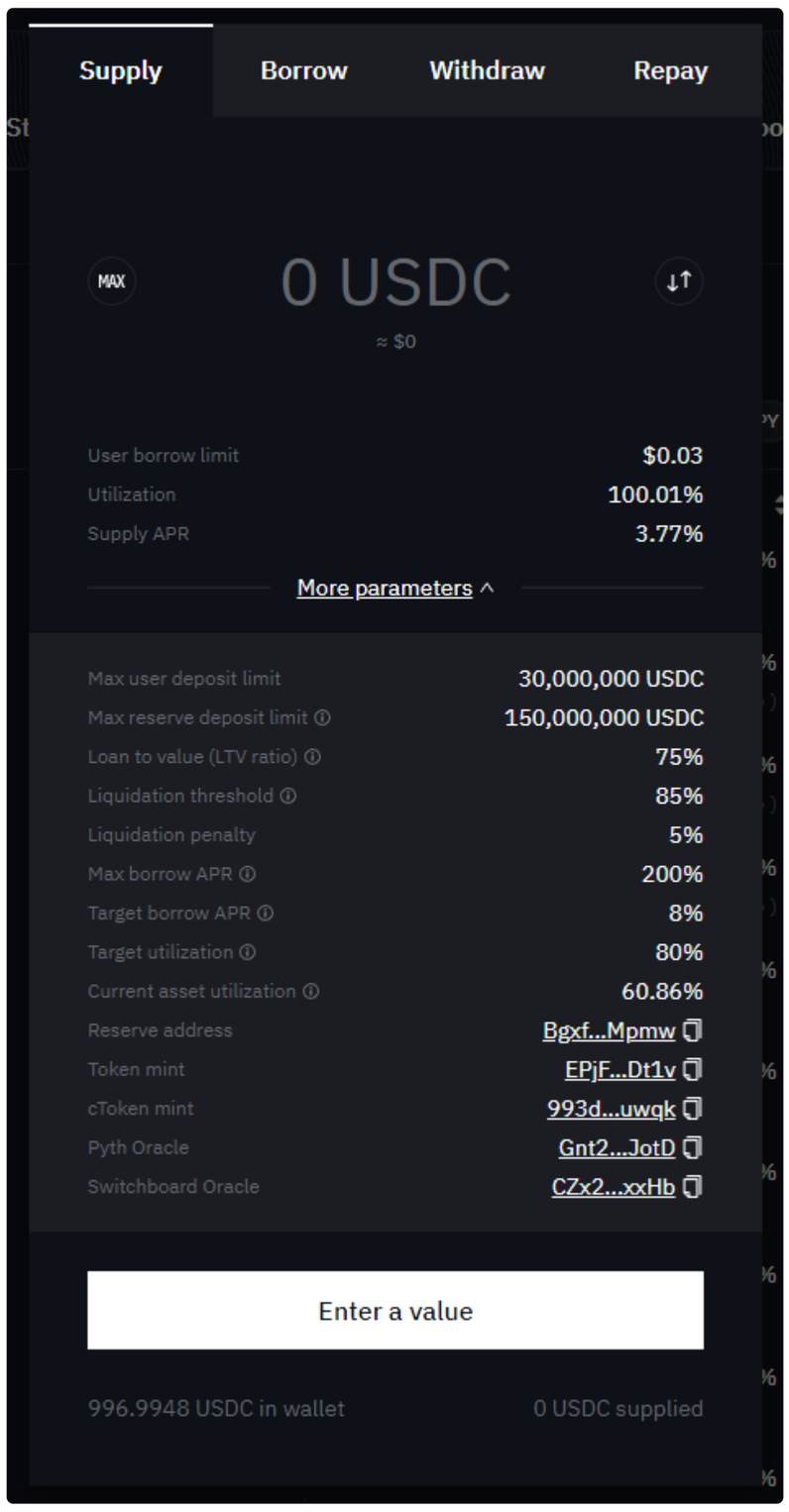
Liquidations are important to maintain pool health and keep deposits safe. It is the pool creator’s responsibility to ensure the health of these pools and ensuring liquidators are operating in the pool.

As it is possible for anyone to create a permissionless pool and list it on Solend, it is important for users to take extra caution. In the event of bad debt or wrongful liquidations, Solend will not be able to compensate users. Solend's team have built an open sourced liquidator here, which should serve as a starting point for you.

Feel free to reach out on Discord if you have questions!

Configs

At pool creation, creators can specify the configs for their pool. Some examples include Loan-To-Value or Max Borrow APRs. These configs will be displayed on the supply/borrow pages; be sure to review them before depositing or borrowing!



Verify the configs and oracles here

Fees

There’s a fee of 100 SLND to create a Permissionless Pool. This fee serves to prevent spam pools from being created. In return, pool creators get 20% of borrow fees according to the host fee priority here.

Integration Guide

Introduction

Solend has 3 types of accounts: lending market, reserve, and obligation.

There's only one instance of a lending market for now. There's a one-to-one mapping from listed asset to reserve. There's a one-to-one mapping from user to obligations (which tracks user's collateral/borrows).

Rust definitions are here.

Calculating TVL from on-chain data example

https://github.com/solendprotocol/tvl-calculator-example

Deriving obligation address

```
const seed = LENDING_MARKET_ADDRESS.slice(0, 32);
const obligationAddress = await PublicKey.createWithSeed(
  new PublicKey(address),
  seed,
  new PublicKey(SOLEND_PROGRAM_ID),
);
```

Calculating APY

The following typescript snippet is extracted from our codebase:

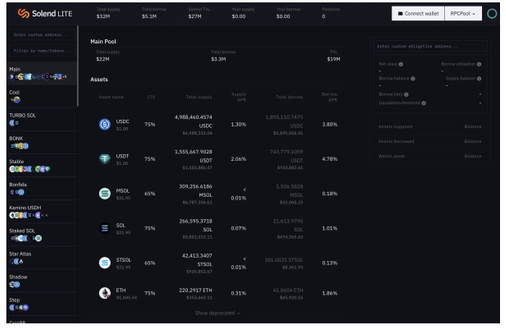
```
export const calculateSupplyAPY = (reserve: Reserve) => {
  const currentUtilization = calculateUtilizationRatio(reserve);
  const borrowAPY = calculateBorrowAPY(reserve);
  return currentUtilization * borrowAPY;
};

export const calculateUtilizationRatio = (reserve: Reserve) => {
  const borrowedAmount = reserve.liquidity.borrowedAmountWads
    .div(new BN(`1${''.padEnd(18, '0')}`))
    .toNumber();
  const availableAmount = reserve.liquidity.availableAmount.toNumber();
  const currentUtilization =
    borrowedAmount / (availableAmount + borrowedAmount);
  return currentUtilization;
};

export const calculateBorrowAPY = (reserve: Reserve) => {
  const currentUtilization = calculateUtilizationRatio(reserve);
  const optimalUtilization = reserve.config.optimalUtilizationRate / 100;
  let borrowAPY;
  if (optimalUtilization === 1.0 || currentUtilization < optimalUtilization) {
    const normalizedFactor = currentUtilization / optimalUtilization;
    const optimalBorrowRate = reserve.config.optimalBorrowRate / 100;
    const minBorrowRate = reserve.config.minBorrowRate / 100;
    borrowAPY =
      normalizedFactor * (optimalBorrowRate - minBorrowRate) + minBorrowRate;
  } else {
    const normalizedFactor =
      (currentUtilization - optimalUtilization) / (1 - optimalUtilization);
    const optimalBorrowRate = reserve.config.optimalBorrowRate / 100;
    const maxBorrowRate = reserve.config.maxBorrowRate / 100;
    borrowAPY =
      normalizedFactor * (maxBorrowRate - optimalBorrowRate) +
      optimalBorrowRate;
  }

  return borrowAPY;
};
```

Additionally, we have a few other resources for developers:

- SDK
- APIs
-

Solend LITE

Solend LITE is a lightweight open-source client to the Solend program.



It is a great starting point to building with Solend!

1. Checkout the code to see how Solend SDK is used
2. Clone it and try self hosting your own Solend client
3. Fork it and use it to bootstrap your Solend integration

Check out the code here!

Liquidators

We rely on third party liquidators, which means anyone can become a liquidator.

GitHub - solendprotocol/liquidator: open source version of a liquidation bot running against …

GitHub

Get started at the link above, and come introduce yourself in our Discord's #dev-intro for the latest updates.

Flash Loans

Currently, this version of flash loan implementation for Solend exists, but functionality is limited due to the reentracy of Solana transactions, where Program A calls Program B, and etc.

solana-program-library/processor.rs at master · solendprotocol/solana-program-library
GitHub

We are currently working on a working flash loan program and the mitigation for it, so that it can be used safely, and will document it here when it's ready.

https://github.com/solendprotocol/solana-program-library/pull/70

Developer Referral Fee

On Solend, 80% of fees are collected as a protocol fee with 20% earmarked as a host fee.

The host fee goes to: (in order of priority)

- Users with 1,000 SLND self-referring
- Referrals from another address
- Permissionless pool creators
- Alternative UIs built on Solend smart contracts
- Solend

When calling the borrow function on the Solend smart contracts, a client may pass in an address to receive the 20% host fee. This is applicable to DeFi strategies integrating Solend (discount off fee), or alternative UIs to Solend's (pass in an address to receive some fees).

This is an incentive to encourage developers to build on top of Solend.

solana-program-library/processor.rs at mainnet · solendprotocol/solana-program-library
GitHub

Resources and FAQ

Resources:

- https://api.solend.fi
- https://sdk.solend.fi

Decimal places?

- Token decimals (find using Solscan)
- Wads: 10^18
- Token decimals for external rewards: 10^36